Fast algorithms for the solution of scattering problems by Boundary Element Methods

Stéphanie Chaillat-Loseille

Laboratoire POEMS (UMR 7231 CNRS-INRIA-ENSTA), ENSTA, France
stephanie.chaillat@ensta-paristech.fr

Ecole Doctorale SIE, Méthodes Numériques Avancées en Mécanique
Jeudi 12 Mars 2015
1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
4. Fast Multipole accelerated BEM
5. Algebraic fast BEMs based on low rank approximations
1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
4. Fast Multipole accelerated BEM
5. Algebraic fast BEMs based on low rank approximations
Definition of the problem

Helmholtz equation

\[ \Delta u + k^2 u = 0, \quad x \in \Omega^\pm, \quad k = \frac{\omega}{c} \]

Dirichlet or Neumann data

\[ u|_\Gamma = u_d \quad \text{or} \quad (\frac{\partial u}{\partial n})|_\Gamma = u_n \]

Sommerfeld radiation condition for exterior problems

\[ \lim_{r \to \infty} r \left( \frac{\partial u}{\partial r} - ik u \right) = 0 \]
Boundary Integral Representation for Helmholtz

Green’s function for the full-space: solution in the sense of distributions

\[ \Delta_x G + k^2 G = -\delta(x, y); \quad G(r) = \frac{e^{ikr}}{4\pi r} \]

Green’s formulas: any solution to Helmholtz equation can be represented as a linear combination of single- and double-layer potentials.

Single and Double layer potentials for \((\phi, \psi) \in H^{1/2}(\Gamma) \times H^{-1/2}(\Gamma)\)

\[ S\phi(x) := \int_{\Gamma} G(x-y)\phi(y)dS_y, \quad D\psi(x) := \int_{\Gamma} \frac{\partial G}{\partial n(y)}(x-y)\psi(y)dS_y, \quad x \notin \Gamma \]

- Solutions of the Helmholtz equation if \(x \notin \Gamma\)
- Satisfy Sommerfeld radiation condition
- \(S\phi\) continuous in \(\mathbb{R}^3\); \(D\psi\) discontinuity at \(\Gamma\)
Boundary Integral Representation for Helmholtz

\[ \Delta u^-(x) + k^2 u^-(x) = 0, \quad x \in \Omega^-; \]

\[ \begin{cases} \Delta u^+(x) + k^2 u^+(x) = 0, & x \in \Omega^+, \\
\text{outgoing wave.} & \end{cases} \]

Trace operators:

\[ \gamma_j^\pm : H^1_\pm(\Delta) \rightarrow H^{1/2-j}(\Gamma), \]

\[ \gamma_j^\pm = \partial_n^j u^\pm|_\Gamma. \]

Integral representation formula

\[ S([\partial_n u(y)]_\Gamma)(x) - D([u(y)]_\Gamma)(x) = \begin{cases} u^+(x), & x \in \Omega^+ \\
u^-(x), & x \in \Omega^- \end{cases}. \]

with \([u]_\Gamma := \gamma_0^- u^- - \gamma_0^+ u^+\) and \([\partial_n u]_\Gamma := \gamma_1^- u^- - \gamma_1^+ u^+.\]

---

Calderon relations

Four elementary boundary integral operators expressed for \(x \in \Gamma\)

\[
S\phi(x) := \int_{\Gamma} G(x - y) \phi(y) dS_y, \quad D\psi(x) := \int_{\Gamma} \frac{\partial G}{\partial n(y)} (x - y) \psi(y) dS_y,
\]

\[
D'\phi(x) := \int_{\Gamma} \frac{\partial G}{\partial n(x)} (x - y) \phi(y) dS_y, \quad N\psi(x) := \int_{\Gamma} \frac{\partial^2 G(x - y)}{\partial n(x) \partial n(y)} \psi(y) dS_y.
\]

Traces of the single and double layer operators

\[
\gamma_0^\pm S = S, \quad \gamma_0^\pm D = \pm \frac{I}{2} + D, \quad \gamma_1^\pm S = \mp \frac{I}{2} + D', \quad \gamma_1^\pm D = N.
\]

Calderon relations

\[
DS = SD', \quad ND = D'N, \quad D^2 - SN = \frac{I}{4}, \quad D'^2 - NS = \frac{I}{4}.
\]

Direct integral formulations Helmholtz: Dirichlet

\[
S([\partial_n u(y)]_\Gamma)(x) - D([u(y)]_\Gamma)(x) = \begin{cases} 
    u^+(x), & x \in \Omega^+ \\
    u^-(x), & x \in \Omega^- 
\end{cases}.
\]

Single-layer potential representation of total field: \((u^-, u^+) = (-u^{inc}, u^+)\)

**EFIE:** apply the trace operator \(\gamma^-\)
Direct integral formulations Helmholtz: Dirichlet

\[ S([\partial_n u(y)]_{\Gamma})(x) - D([u(y)]_{\Gamma})(x) = \begin{cases} u^+(x), & x \in \Omega^+ \\ u^-(x), & x \in \Omega^- \end{cases}. \]

Single-layer potential representation of total field: \((u^-, u^+) = (-u^{inc}, u^+)\)

\([u]_{\Gamma} = 0 \text{ and } [\partial_n u]_{\Gamma} = -\gamma_1^+(u^+ + u^{inc}) := -\gamma_1^+ w|_{\Gamma} := p.\]

\[ Sp(x) + u^{inc}(x) = \begin{cases} 0, & x \in \Omega^- \\ w(x), & x \in \Omega^+ \end{cases}. \]

**EFIE:** apply the trace operator \(\gamma_0^-\)
Direct integral formulations Helmholtz: Dirichlet

\[ S([\partial_n u(y)]_\Gamma)(x) - D([u(y)]_\Gamma)(x) = \begin{cases} u^+(x), & x \in \Omega^+ \\ u^-(x), & x \in \Omega^- \end{cases}. \]

Single-layer potential representation of total field: \((u^-, u^+) = (-u^{inc}, u^+)\)

\([u]_\Gamma = 0 \quad \text{and} \quad [\partial_n u]_\Gamma = -\gamma_1^+ (u^+ + u^{inc}) := -\gamma_1^+ w|_\Gamma := p.\]

\[ Sp(x) + u^{inc}(x) = \begin{cases} 0, & x \in \Omega^- \\ w(x), & x \in \Omega^+ \end{cases}. \]

**EFIE:** apply the trace operator \(\gamma_0^{-}\)

\[ Sp = -\gamma_0^+ u^{inc}, \text{ on } \Gamma. \]

**MFIE:** apply the trace operator \(\gamma_1^{-}\)
Direct integral formulations Helmholtz: Dirichlet

\[ S(\partial_n u(y)\big|_\Gamma)(x) - D(u(y)\big|_\Gamma)(x) = \begin{cases} u^+(x), & x \in \Omega^+ \\ u^-(x), & x \in \Omega^- \end{cases}. \]

Single-layer potential representation of total field: \((u^-, u^+) = (-u^{inc}, u^+)\)

\([u]_\Gamma = 0 \text{ and } [\partial_n u]_\Gamma = -\gamma_1^+(u^+ + u^{inc}) := -\gamma_1^+ w|_\Gamma := p.\)

\[ Sp(x) + u^{inc}(x) = \begin{cases} 0, & x \in \Omega^- \\ w(x), & x \in \Omega^+ \end{cases}. \]

**EFIE:** apply the trace operator \(\gamma_0^-\)

\[ Sp = -\gamma_0^+ u^{inc}, \text{ on } \Gamma. \]

**MFIE:** apply the trace operator \(\gamma_1^-\)

\[ \left( \frac{I}{2} + D' \right)p = -\gamma_1^+ u^{inc}, \text{ on } \Gamma. \]

**CFIE:** apply the trace operator \(\gamma_1^- + \eta \gamma_0^-\)
Direct integral formulations Helmholtz: Dirichlet

\[ S([\partial_n u(y)]_\Gamma)(x) - D([u(y)]_\Gamma)(x) = \begin{cases} 
  u^+(x), & x \in \Omega^+ \\
  u^-(x), & x \in \Omega^- 
\end{cases} \]

Single-layer potential representation of total field: \((u^-, u^+) = (-u^{inc}, u^+)\)

\[ [u]_\Gamma = 0 \text{ and } [\partial_n u]_\Gamma = -\gamma^+_1 (u^+ + u^{inc}) := -\gamma^+_1 w|_\Gamma := p. \]

\[ Sp(x) + u^{inc}(x) = \begin{cases} 
  0, & x \in \Omega^- \\
  w(x), & x \in \Omega^+ 
\end{cases} \]

**EFIE:** apply the trace operator \(\gamma^-_0\)

\[ Sp = -\gamma^+_0 u^{inc}, \text{ on } \Gamma. \]

**MFIE:** apply the trace operator \(\gamma^-_1\)

\[ (\frac{I}{2} + D') p = -\gamma^+_1 u^{inc}, \text{ on } \Gamma. \]

**CFIE:** apply the trace operator \(\gamma^-_1 + \eta \gamma^-_0\)

\[ \left\{ (\frac{I}{2} + D') + \eta S \right\} p = -(\gamma^+_1 u^{inc} + \eta \gamma^+_0 u^{inc}), \text{ on } \Gamma. \]
Indirect integral formulations Helmholtz: Dirichlet

- Sol. expressed in terms of a source density function defined on $\Gamma$
- Unknowns are non-physical quantities
- Exterior field seek as a superposition of single- and double- layer potentials acting on a fictitious surface density $\psi$

Burton-Miller/ Brackage-Werner indirect integral formulations

$$u^+(x) = (D + \eta S)\psi(x), \quad x \in \Omega^+.$$  

Corresponding Integral Equation
Indirect integral formulations Helmholtz: Dirichlet

- Sol. expressed in terms of a source density function defined on $\Gamma$
- Unknowns are non-physical quantities
- Exterior field seek as a superposition of single- and double-layer potentials acting on a fictitious surface density $\psi$

Burton-Miller/ Brackage-Werner indirect integral formulations

$$u^+(x) = (D + \eta S)\psi(x), \quad x \in \Omega^+.$$  

Corresponding Integral Equation

$$\left\{(\frac{I}{2} + D) + \eta S\right\} \psi = -\gamma_0^+ u^{inc}, \quad \text{on } \Gamma.$$

Uniqueness: This integral has a unique solution if and only if $\Im(\eta) > 0$. Almost optimal value: $\eta = ik$. 
Direct integral formulations Helmholtz: Neumann

\[ S([\partial_n u(y)]\Gamma)(x) - D([u(y)]\Gamma)(x) = \begin{cases} u^+(x), & x \in \Omega^+ \\ u^-(x), & x \in \Omega^- \end{cases} \]

Double-layer potential representat. of total field: \((u^-, u^+) = (-u^{inc}, u^+)\)

\[ [u]\Gamma = -\gamma_0^+(u^+ + u^{inc}) := -\gamma_0^w := \phi \text{ and } [\partial_n u]\Gamma = 0. \]

\[ D\phi(x) + u^{inc}(x) = \begin{cases} 0, & x \in \Omega^- \\ w(x), & x \in \Omega^+ \end{cases} \]

**EFIE:** apply the trace operator \(\gamma_1^-\)

\[ N\phi = -\gamma_1^+ u^{inc}, \text{ on } \Gamma. \]

**MFIE:** apply the trace operator \(\gamma_0^-\)

\[ (-\frac{I}{2} + D)\phi = -\gamma_0^+ u^{inc}, \text{ on } \Gamma. \]

**CFIE:** apply the trace operator \(\gamma_0^- + \eta \gamma_1^-\)

\[ \left\{ \left(-\frac{I}{2} + D\right) + \eta N \right\} \phi = -(\gamma_0^+ u^{inc} + \eta \gamma_1^+ u|^{inc}), \text{ on } \Gamma. \]
Indirect integral formulations Helmholtz: Neumann

- Sol. expressed in terms of a source density function defined on $\Gamma$
- Unknowns are non-physical quantities
- Exterior field seek as a superposition of single- and double- layer potentials acting on a fictitious surface density $\varphi$

Burton-Miller/ Brackage-Werner indirect integral formulations

$$u^+(x) = (S + \eta D)\varphi(x), \quad x \in \Omega^+.$$ 

Corresponding Integral Equation
Indirect integral formulations Helmholtz: Neumann

- Sol. expressed in terms of a source density function defined on $\Gamma$
- Unknowns are non-physical quantities
- Exterior field seek as a superposition of single- and double-layer potentials acting on a fictitious surface density $\varphi$

Burton-Miller/ Brackage-Werner indirect integral formulations

$$ u^+(x) = (S + \eta D)\varphi(x), \quad x \in \Omega^+. $$

Corresponding Integral Equation

$$ \left\{ ( - \frac{I}{2} + D') + \eta N \right\} \varphi = -\gamma_1^+ u^{inc}, \text{ on } \Gamma. $$

**Uniqueness:** This integral has a unique solution if and only if $\Im(\eta) > 0$.

Almost optimal value: $\eta = 1/(ik)$. 

11 / 141  S. Chaillat-Loseille  Fast algorithms for solving scattering pb by BEM
Another formulation

\[ S([\partial_n u(y)]_\Gamma)(x) - D([u(y)]_\Gamma)(x) = \begin{cases} u^+(x), & x \in \Omega^+ \\ u^-(x), & x \in \Omega^- \end{cases}. \]

Extension of the trace by zéro: \((u^-, u^+) = (0, u^+)\)

\[ [u]_\Gamma = -\gamma^+_0 u^+ \text{ and } [\partial_n u]_\Gamma = -\gamma^+_1 u^+. \]

\[ -S(\gamma^+_1 u^+)(x) + D(\gamma^+_0 u^+)(x) = \begin{cases} u^+(x), & x \in \Omega^+ \\ 0, & x \in \Omega^- \end{cases}. \]

Corresponding integral equation: apply the trace operator \(\gamma^+_0\)

\[ ( - \frac{I}{2} + D ) \gamma^+_0 u^+ = S \gamma^+_1 u^+, \text{ on } \Gamma. \]

\(\Rightarrow\) Helmholtz representation formula leads to infinite # of integral equations
How to derive others Green’s functions?

Method of images for half-space problems

\[ G_{HS}(\mathbf{x}, \mathbf{y}) = G(\mathbf{x}, \mathbf{y}) + G(\mathbf{\tilde{x}}, \mathbf{y}); \]

\[ G_{HS}(\mathbf{x}, \mathbf{y}) = G(\mathbf{x}, \mathbf{y}) - G(\mathbf{\tilde{x}}, \mathbf{y}); \]

Neumann BC on free surface

Dirichlet BC on free surface
Extension to elastic scattering

Navier Equation: isotropic, homogeneous and linear elasticity

- Equation of motion: \( \nabla \cdot \sigma + \rho \omega^2 u = 0 \);
- Strain-displacement equation: \( \varepsilon = \frac{1}{2} [\nabla u + (\nabla u)^T] \)
- Constitutive equation: \( \sigma = C : \varepsilon \)

\[
\mu \Delta u + (\lambda + \mu) \nabla \text{div} u + \rho \omega^2 u = 0
\]

Lamé decomposition: \( u = \nabla \phi + \nabla \times \psi = u_p + u_s \)

\[
\Delta \phi + k_p^2 \phi = 0, \quad \Delta \psi + k_s^2 \psi = 0
\]

Kupradze-Sommerfeld radiation condition for exterior problems

\[
\lim_{r \to \infty} r \left( \frac{\partial u_p}{\partial r} - i k_p u_p \right) = 0, \quad \lim_{r \to \infty} r \left( \frac{\partial u_s}{\partial r} - i k_s u_s \right) = 0, \quad r = |x|
\]

\[
k_p^2 = \rho \omega^2 (\lambda + 2\mu)^{-1} \quad \text{and} \quad k_s^2 = \rho \omega^2 \mu^{-1}
\]
Boundary Integral Representation for elastic scattering

Green’s function for the full-space: distributions

\[ U(x, y) = \frac{1}{\rho \omega^2} \left( \text{curl} \ curl_x \ \{ G(k_s, x - y) I_3 \} - \nabla_x \text{div}_x \ \{ G(k_p, x - y) I_3 \} \right) \]

Potential theory: any solution to Navier equation can be represented as a linear combination of a single- and a double-layer potentials.

Single and Double layer potentials

\[ S\phi(x) := \int_{\Gamma} U(x, y) \phi(y) dS_y, \quad D\psi(x) := \int_{\Gamma} [T_y U(x, y)]^T \psi(y) dS_y, \quad x \notin \Gamma \]

\( T_y \) applied to each column; \( T = 2\mu \frac{\partial}{\partial n} + \lambda n \text{div} + \mu n \times \text{curl}. \)

- Solutions of the Navier equation if \( x \notin \Gamma \)
- Satisfy Kupradze radiation condition
Outline

1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
4. Fast Multipole accelerated BEM
5. Algebraic fast BEMs based on low rank approximations
Outline of the boundary element method

Two main steps

1. Solve the boundary integral equation on the boundary
   - Linear system to solve
   - Unknowns only on the boundary

2. Invoke the integral representation for the evaluation of the quantities at interior points
   - Cost reduced to a matrix-vector multiplication

Example: EFIE with Dirichlet Boundary Condition
Outline of the boundary element method

Two main steps

1. Solve the boundary integral equation on the boundary
   - Linear system to solve
   - Unknowns only on the boundary

2. Invoke the integral representation for the evaluation of the quantities at interior points
   - Cost reduced to a matrix-vector multiplication

Example: EFIE with Dirichlet Boundary Condition

1. Solve the boundary integral equation

\[ Sp = -\gamma_0^+ u^{inc}, \text{ on } \Gamma. \]

2. Apply the boundary integral representation

\[ u^+(x) = Sp(x), \ x \in \Omega^+. \]

3. Normal derivative of the trace of \( u^+(x) \):

\[ p = -\gamma_1^+(u^+ + u^{inc}) \]

\[ \partial_n u^+_{\Gamma} = -p - \partial_n u^{inc}, \text{ on } \Gamma. \]
Discretization methods

First step: How do we discretize integrals? \( u(x) = \int_{\Gamma} G(x, y)p(y)\,dS_y \)

**Nyström methods.** Direct evaluation of the kernel using (non-standard) quadrature rules. The solution is \( p(y_j) \).

\[
u(x_i) \approx \sum_{j \neq i} G(x_i, y_j)p(y_j)\omega_j
\]

**Collocation methods.** Expand \( p(y) = \sum_j p_j v_j(y) \) in some basis and evaluate point wise. The solution is \( p_j \).

\[
u(x_i) \approx \sum_j p_j \int_{\Gamma} G(x_i, y)v_j(y)\,dS_y
\]

**Galerkin methods.** Expand \( p(y) = \sum_j p_j v_j(y) \) and \( u(x) = \sum_j u_j v_j(x) \). Use of a variational formulation. The solution is \( p_j \).

\[
\sum_j u_j \langle v_i, v_j \rangle = \sum_j p_j \langle v_i, T_G v_j \rangle \quad \text{with} \quad (T_G \psi)(x) = \int_{\Gamma} G(x, y)\psi(y)\,dS_y
\]
Discretization methods

- **Collocation methods:**
  \[ u(x_i) \approx \sum_j p_j \int_{\Gamma} G(x_i, y)v_j(y)\,dS_y \]

- **Galerkin methods:**
  \[ \sum_j u_j \langle v_i, v_j \rangle = \sum_j p_j \langle v_i, T_G v_j \rangle \quad \text{with} \quad (T_G \psi)(x) = \int_{\Gamma} G(x, y)\psi(y)\,dS_y \]

What is the difference between the collocation and Galerkin methods?

- Extra computational cost for Galerkin (additional integration)
- More mathematical background to justify the Galerkin approach

⇒ In the following, we will consider only the collocation approach.
Discretization methods

- Collocation methods:
  \[ u(x_i) \approx \sum_j p_j \int_{\Gamma} G(x_i, y)v_j(y)\,dS_y \]

- Galerkin methods:
  \[ \sum_j u_j \langle v_i, v_j \rangle = \sum_j p_j \langle v_i, T_G v_j \rangle \quad \text{with} \quad (T_G \psi)(x) = \int_{\Gamma} G(x, y)\psi(y)\,dS_y \]

What is the difference between the collocation and Galerkin methods?
- Extra computational cost for Galerkin (additional integration)
- More mathematical background to justify the Galerkin approach

⇒ In the following, we will consider only the collocation approach.
Mesh of the boundary

\[ u(x_i) \approx \sum_j p_j \int_\Gamma G(x_i, y)v_j(y)dS_y \]

Discretization of the geometry: Partition of the boundary \( \Gamma \) into elements (possibly curvilinear and with curvilinear edges): \( \Gamma = \bigcup_{e=1}^{N_e} E_e \)

\[ \sum_j p_j \int_\Gamma G(x_i, y)v_j(y)dS_y = \sum_j p_j \sum_{e=1}^{N_e} \int_{E_e} G(x_i, y)v_j(y)dS_y \]

Remark for scattering problems: about 10 points per \( \lambda = \frac{2\pi}{k} \)

Description of the elements with nodes and shape functions
All the shape functions on a mesh would be different.

Introduction of an analytic description of each element

\[ \xi \in \Delta_e \rightarrow y(\xi) = \sum_{m=1}^{N_e} v_m(\xi) y^m \in E_e \]

\( \Delta_e \) is the reference element

\( y^m \): nodes and \( v_p(\xi) \): shape functions

\[ v_p(\xi^q) = \delta_{pq}, \quad \sum_{m=1}^{N_e} v_q(\xi) = 1 \]

\[ \Rightarrow \text{2 important matrices: nodal coordinates and connectivity tables} \]

Example: three-noded triangular

- Nodes: vertices of the triangles
- Mapping onto the reference element
  \[ y = \xi_1 y^1 + \xi_2 y^2 + (1 - \xi_1 - \xi_2) y^3 \]
- Natural basis of tangent plane to \( E_e \)
  \[ a_1 = y^1 - y^3, \quad a_2 = y^2 - y^3 \]
- Jacobian of the mapping
  \[ J(\xi) = |a_1 \wedge a_2| \]
- Unit normal
  \[ n = \frac{1}{J} (a_1 \wedge a_2) \]
- Area of a differential surface element
  \[ dS_y = J(\xi) d\xi \]
- Integration
  \[ \int_{E_e} f(y) dS_y = \int_{\Delta_e} f(\xi) J(\xi) d\xi \]
Discretization of the unknowns

- Similar description of the unknowns with the mapping
  \[ \xi \in \Delta_e \rightarrow y(\xi) \in E_e \]

- Approximation of the unknowns
  \[ p(y) = \sum_{j=1}^{NI(e)} p^j M_j(\xi), \quad \xi \in \Delta_e \]

- \( p^j = p(z^j) \) with \( z^j \): interpolation nodes and \( M_j \) interpolation function

\[ M_p(\xi^q) = \delta_{pq}, \quad \sum_{m=1}^{NI(e)} M_q(\xi) = 1 \]

Isoparametric representation of the boundary and unknowns
- Geometrical nodes are used as interpolation nodes
- Shape and interpolation functions are the same
Illustration on the EFIE for Dirichlet problems

\[ \int_{\Gamma} G(x - y) p(y) \, ds_y = -u^{inc}(x), \quad x \in \Gamma. \]

- Introduction of \( N \) collocation points on the boundary

\[ \int_{\Gamma} G(x^i - y) p(y) \, ds_y = -u^{inc}(x^i) := b_i \]

- Discretization of the boundary

\[ \sum_{e=1}^{N_e} \int_{E_e} G(x^i - y) p(y) \, ds_y = -u^{inc}(x^i) \]

- Interpolation of the unknown: on \( E_e \), \( p(y) = \sum_{j=1}^{N_e} p^j v_j(\xi) \)

\[ \sum_{e=1}^{N_e} \sum_{j=1}^{N_I(e)} p^j \int_{E_e} G(x^i - y) v_j(\xi) \, ds_y = -u^{inc}(x^i) \]
Computation of the system entries

\[ Ap = b \quad (A \in \mathbb{C}^{N \times N}, \ b \in \mathbb{C}^{N}) \]

- \( A_{ij} \): collocation node \( i \) and interpolation node \( j \)
- The vector \( p \) gives the nodal values
- Matrix \( A \): square, fully-populated, invertible, non-symmetric

⇒ What is the next step? The main difficulty?
Computation of the system entries

\[ Ap = b \quad (A \in \mathbb{C}^{N\times N}, b \in \mathbb{C}^N) \]

- \( A_{ij} \): collocation node \( i \) and interpolation node \( j \)
- The vector \( p \) gives the nodal values
- Matrix \( A \): square, fully-populated, invertible, non-symmetric

⇒ What is the next step? The main difficulty?

Evaluation of element integrals

- If \( x^i \notin E_e \): Non-singular element integral \( \rightarrow \) Gaussian quadrature
- If \( x^i \in E_e \): Singular element integral
  - Weakly singular integrals \( O(r^{-1}) \): polar coordinate transform
  - Strongly singular integrals \( O(r^{-2}) \): regularization techniques
  - For simple elements and interpolations: analytic singular integration available
Gaussian quadratures

Gaussian quadrature on $[-1, 1]$

\[
\int_{-1}^{1} f(x) \, dx \approx \sum_{i=1}^{N} w_i f(x_i)
\]

- $x_i$: points; $w_i$: weights
- $N$-point formula is exact for all polynomials of degree $2N - 1$

Gaussian quadrature on $[a, b]$: change of variable

\[
\int_{a}^{b} f(x) \, dx = \frac{b - a}{2} \int_{-1}^{1} f\left(\frac{b + a}{2} + \frac{b - a}{2} \xi \right) \, d\xi
\]

Gaussian quadrature on $[-1, 1]^2$: product rule

\[
\int_{-1}^{1} \int_{-1}^{1} f(x, y) \, dx \, dy \approx \sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j f(x_i, x_j)
\]
Evaluation of element integrals: nonsingular

Integration on a triangle: $\mathcal{T}: \xi_1 \geq 0, \xi_2 \geq 0, 1 - \xi_1 - x_2 \geq 0$

- Mapping onto the unit sphere
  \[
  \xi_1 = 0.25(1 + u)(1 - v), \quad \xi_2 = 0.5(1 + v)
  \]

\[
\int_{\mathcal{T}} f(x, y) \, dx \, dy = \int_{-1}^{1} \int_{-1}^{1} g(u, v) \frac{1}{8}(1 - v) \, du \, dv \approx \frac{1}{8} \sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j g(x_i, x_j)(1 - y_j)
\]

Integration on a triangle: specific quadrature rule

\[
\int_{\mathcal{T}} f(x, y) \, dx \, dy \approx \sum_{i=1}^{N} w_i^T f(x_i^T, y_i^T)
\]

- Points and weights on the reference triangle
  - $N = 3$: $x_i^T = 0.16666667$, $y_i^T = 0.16666667$, $w_i^T = 0.16666667$

What is the relation between the systems obtained with the Galerkin
and the collocation methods?

- Collocation

\[
\begin{bmatrix}
    b_1 \\
    \vdots \\
    b_N
\end{bmatrix} =
\begin{bmatrix}
    G(x^i, y^j) & w_1 & v_1(y_1) & \ldots & v_N(y_1) \\
      & \vdots & \vdots & \vdots & \vdots \\
    w_{N_g} & v_1(y_{N_g}) & \ldots & v_N(y_{N_g})
\end{bmatrix}
\begin{bmatrix}
    p_1 \\
    \vdots \\
    p_N
\end{bmatrix}
\]

\[b = \{G\} w \{v\} p\]

- Galerkin

\[
\tilde{p} \{v\} w b^T = \tilde{p} \{v\} w \{G\}^T w^T \{v\}^T p^T
\]
Solution of the linear system

\[ N: \text{number of BE DOFs} \]

\[ \mathbf{A} \mathbf{p} = \mathbf{b} \]

\[ \begin{cases} \mathbf{A} \in \mathbb{C}^{N \times N} \\ \mathbf{p} \in \mathbb{C}^N \\ \mathbf{b} \in \mathbb{C}^N \end{cases} \]

- Matrix \( \mathbf{A} \): square, fully-populated, invertible, non-symmetric
- Direct solvers (LU factorization, ...)
- Iterative solvers (GMRES, ...)
Direct Solvers

LU factorization

- Gaussian elimination applied to arbitrary system requires $O(N^3)$

- Replace the initial system $Ax = b$ by the factorization form $L U x = b$ with $L$ lower triangular and $U$ upper triangular

  - Two-step process: $y = U x$ with $L y = b$

  - Number of operations to solve the system reduces to $O(N^2)$

  - But the factorization costs $O(N^3)$
Direct Solvers

LU factorization

- Gaussian elimination applied to arbitrary system requires $O(N^3)$
- Replace the initial system $Ax = b$ by the factorization form $LUx = b$ with $L$ lower triangular and $U$ upper triangular
  - Two-step process: $y = Ux$ with $Ly = b$
  - Number of operations to solve the system reduces to $O(N^2)$
  - But the factorization costs $O(N^3)$

Advantages and drawbacks?

- Pros: robust, accurate, multi RHS
- Cons: $O(N^2)$ memory and $O(N^3)$ CPU

Recent works on direct solvers

- MUMPS: a MUltifrontal Massively Parallel sparse direct Solver
- But in our case the matrix is fully-populated
Ax = b

- \( A \): square invertible matrix
- \( x_0 \): initial iterate; \( r_0 = Ax_0 - b \): initial residual
- Conjugate Gradient and GMRES minimize at kth iteration some measure of error over the affine space

\[ x_0 + \mathcal{K}_k(A, r_0) \]

Definition (Krylov space of order \( k \) generated by a vector \( v \))

\[ \mathcal{K}_k(A, v) = \text{span}\{v, Av, A^2v, \ldots, A^{k-1}v\} \]

- Conjugate Gradient: to solve symmetric positive definite systems

Orthogonal basis of Krylov space

- We need a basis of $K_k$
- $(r_0, Ar_0, A^2r_0, \ldots, A^{k_{max}}r_0)$ is not an option: vectors numerically dependent $\Rightarrow$ Arnoldi basis

Arnoldi process

 Initialization
\[
\begin{align*}
  r_0 &= Ax_0 - b \\
  v_1 &= r_0 / \|r_0\| 
\end{align*}
\]

Derivation of the vector $j + 1$ in the Arnoldi basis
\[
\begin{align*}
  w &= Av_j \\
  \text{for } i = 1 \ldots j & \text{ do} \\
  & \quad \alpha_i = (w, v_i) \\
  & \quad w = w - \alpha_i v_i \\
  \text{end for} \\
  v_{j+1} &= w / \|w\| 
\end{align*}
\]

$\Rightarrow$ Only matrix-vector products used
Arnoldi process

\[ r_0 = Ax_0 - b \]
\[ v_1 = r_0 / \| r_0 \| \]

for \( j = 1 \ldots k \) do

\[ w = Av_j \]

for \( i = 1 \ldots j \) do

\[ h_{ij} = (w, v_i) \]
\[ w = w - h_{ij} v_i \]

end for

\[ h_{j+1} = \| w \| \]
\[ v_{j+1} = w / h_{j+1} \]

end for

Return \((v_1, \ldots, v_{k_{max}+1})\)

\[ v_{j+1} h_{j+1} = w = Av_j - \sum_{i=1}^{j} h_{ij} v_i \iff Av_j = \sum_{i=1}^{j+1} h_{ij} v_i \]

\[ AV_k = V_{k+1} \bar{H}_k \]
Generalized Minimal RESiduals

- Approximation of solution $Ax = b$ in $x_0 + \mathcal{K}_k$ ($A$ nonsymmetric)

- Orthogonal basis of $\mathcal{K}_k$ build with the Arnoldi’s procedure

$$AV_k = V_k H_k + w_k e_k^T = V_{k+1} \bar{H}_k$$

- Residual:

$$r_k = Ax_k - b = A(x_0 + V_k z_k) - b = V_{k+1} \bar{H}_k z_k - r_0$$

- $V_k^T V_k = I_k \Rightarrow V_{k+1}^T r_k = \bar{H}_k z_k - V_{k+1}^T r_0$

- By construction: $v_1 = r_0/||r_0||$ and $v_j^T r_0 = 0, \forall j > 1$

$$V_{k+1}^T r_k = \bar{H}_k z_k - e_1 ||r_0||$$

- Algorithm based on solution of least square problems

$$\bar{H}_k z_k - e_1 ||r_0||$$
GMRES: principle

1. Calculate $v_k$ with the Arnoldi method
2. Find the $y_k$ which minimizes $\|r_0e_1 - \overline{H}_k y_k\|_2$
3. Compute $x_k = x_0 + V_k y$
4. Repeat if prescribed accuracy is not achieved

Advantages and drawbacks?
- **Pros:** $O(N_{\text{iter}} \times N^2)$ CPU;
- **Cons:** $O(N^2)$ memory; $N_{\text{iter}}$ may be large; only one RHS;
Pros and cons of Boundary Element Methods for waves?

Domain methods (FEM, SEM, ...)
→ Domain mesh
→ Approx. radiation conditions
→ Sparse matrix

BEM
→ Surface mesh (i.e. reduced dim.)
→ Exact radiation conditions
→ Fully-populated matrix

→ BEM adequate for large (unbounded) media, simple (linear) prop
→ Fully-populated BEM influence matrix: severe limiting factor
Computational limitations of standard BEM

Solution of fully-populated matrix equation

- Direct solvers (LU factorization, ...)
  - **Pros**: robust, accurate;
  - **Cons**: $O(N^2)$ memory and $O(N^3)$ CPU

- Iterative solvers (GMRES, ...)
  - **Pros**: $O(N_{\text{iter}} \times N^2)$ CPU;
  - **Cons**: $O(N^2)$ memory; $N_{\text{iter}}$ may be large

($N$: number of BE DOFs)
Computational limitations of standard BEM

Solution of fully-populated matrix equation

- Direct solvers (LU factorization, ...)
  - Pros: robust, accurate;
  - Cons: $O(N^2)$ memory and $O(N^3)$ CPU

- Iterative solvers (GMRES, ...)
  - Pros: $O(N_{\text{iter}} \times N^2)$ CPU;
  - Cons: $O(N^2)$ memory; $N_{\text{iter}}$ may be large

($N$: number of BE DOFs)

Limitations of standard BEM

- High memory cost: assembly of the system matrix $O(N^2)$
- Limited geometric complexity
- Limited (piecewise) heterogeneity
- Limited frequency range (the mesh size depends on $k$)
Computational limitations of standard BEM

Solution of fully-populated matrix equation
- Direct solvers (LU factorization, ...)
  - Pros: robust, accurate;
  - Cons: $O(N^2)$ memory and $O(N^3)$ CPU
- Iterative solvers (GMRES, ...)
  - Pros: $O(N_{\text{iter}} \times N^2)$ CPU;
  - Cons: $O(N^2)$ memory; $N_{\text{iter}}$ may be large

($N$: number of BE DOFs)

Limitations of standard BEM
- High memory cost: assembly of the system matrix $O(N^2)$
- Limited geometric complexity
- Limited (piecewise) heterogeneity
- Limited frequency range (the mesh size depends on $k$)

Towards real-life applications ⇒ Use of Fast BEMs
(FMM, $\mathcal{H}$—matrices, preconditioning techniques, ...)

[37 / 141] S. Chaillat-Loseille

Fast algorithms for solving scattering pb by BEM
Applications in acoustics

Tool for spatial sound rendering through headphones for blind people

Pressure field radiated from the muffler around the car

Surface currents on a A319 illuminated by an antenna

Applications in elastodynamics

Modelling of elastic wave propagation in large/unbounded domains

- Site effects
- Soil-structure interaction
- Computational forward solution method for inverse problems
Numerical verification of theoretical complexity (CPU)

Time-domain results

Diffraction of plane SV-wave by a semispherical sedimentary basin

\[ \mu_1 = 1; \rho_1 = 1; \nu_1 = 1/3; \]
\[ \mu_2 = 1/6; \rho_2 = 2/3; \nu_2 = 1/3; \]
\[ N = 36,033 \]

\( \Omega_1 \)
\( \Omega_2 \)
\( D = 5a \)
plane SV wave
\( \theta = 30^\circ \)
free surface
semi-infinite medium

Time-domain results

Diffraction of plane SV-wave by a semispherical sedimentary basin

\[ \Omega_1 \quad a \quad \Omega_2 \]
\[ D = 5a \]
\[ \text{semi-infinite medium} \]
\[ \text{plane SV wave} \]
\[ \theta = 30^\circ \]

\[ \mu_1 = 1; \rho_1 = 1; \nu_1 = 1/3; \]
\[ \mu_2 = 1/6; \rho_2 = 2/3; \nu_2 = 1/3; \]
\[ N = 36,033 \]

Fast Fourier synthesis of frequency-domain responses

- Incident signal: Ricker wavelet, predominant period \( t_p = 4 \);
- \( 0 < \omega < 3.5 \times 2\pi/t_p \), 32 sample frequencies;
- \( N = 36,033 \) (for lowest frequencies) and \( N = 143,451 \).
Time-domain results

Diffraction of plane SV-wave by a semispherical sedimentary basin

\[ \Omega_1 \]
\[ a \]
\[ D = 5a \]
\[ \Omega_2 \]
\[ \text{semi-infinite medium} \]

\[ \mu_1 = 1; \rho_1 = 1; \nu_1 = 1/3; \]
\[ \mu_2 = 1/6; \rho_2 = 2/3; \nu_2 = 1/3; \]
\[ N = 36,033 \]

Fast Fourier synthesis of frequency-domain responses

- Incident signal: Ricker wavelet, predominant period \( t_p = 4 \);
- \( 0 < \omega < 3.5 \times 2\pi/t_p \), 32 sample frequencies;
- \( N = 36,033 \) (for lowest frequencies) and \( N = 143,451 \).

Time domain surface response: \( u_x \), \( u_x \) (truncated scale)

Accelerating Iterative Solvers

Advantages and drawbacks?

- **Pros:** $O(N_{\text{iter}} \times N^2)$ CPU
- **Cons:** $O(N^2)$ memory; $N_{\text{iter}}$ may be large; only one RHS

**Fast Multipole Method (FMM):** reduction of the complexity

- Based on **iterative** linear equation solvers (GMRES)
- Fast, approximate method for evaluating the linear integral operator (matrix-vector product, called by iterative solver)
- **Analytic** expansions of the kernel
- Kernel dependent or independent; low- or high-frequency

**Preconditioners:** reduction of the number of iterations

\[ MAx = Mb \text{ with } M \approx A^{-1} \]

- Choice of the boundary integral equation
- Algebraic preconditioners
- Analytic preconditioners (derived at the continuous level)
Accelerating Direct Solvers

Advantages and drawbacks?

- **Pros**: robust, accurate, multi RHS, no problem of convergence
- **Cons**: $O(N^2)$ memory and $O(N^3)$ CPU

Low-rank approximations

- Storage of the fully-populated matrix $A$ is $O(N^2)$
- Can we approximate $A$ by a *sparse* matrix $\tilde{A}$?

$\mathcal{H}$ – matrices

- We can decompose $A$ into submatrices that are low rank
- New LU factorization taking into account the structure of $\tilde{A}$
Outline

1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
4. Fast Multipole accelerated BEM
5. Algebraic fast BEMs based on low rank approximations
Computational limitations of standard BEM

Solution of fully-populated matrix equation
- Direct solvers (LU factorization, ...)
  - Pros: robust, accurate;
  - Cons: $O(N^2)$ memory and $O(N^3)$ CPU
- Iterative solvers (GMRES, ...)
  - Pros: $O(N_{\text{iter}} \times N^2)$ CPU;
  - Cons: $O(N^2)$ memory; $N_{\text{iter}}$ may be large

($N$: number of BE DOFs)

Fast Multipole Method (FMM):
- Based on iterative linear equation solvers (GMRES)
- Fast, approximate method for evaluating the linear integral operator (matrix-vector product, called by iterative solver)
How to reduce the costs of the BEM?

BEM discretization $\Rightarrow$ fully-populated system of linear equations

$$Ax = b$$

- Assembly of the matrix system is $O(N^2)$
- Matrix-vector product is $O(N^2)$

How can we reduce the costs?
How to reduce the costs of the BEM?

BEM discretization ⇒ fully-populated system of linear equations

\[ Ax = b \]

- Assembly of the matrix system is \( O(N^2) \)
- Matrix-vector product is \( O(N^2) \)

How can we reduce the costs?

- The system matrix is not assembled: \((A)_{ij}\) is not computed
- The matrix-vector product is approximated
- The complexity of the FMM is \( O(N \log^\alpha N) \) with \( \alpha = 0, 1 \)
1. Principle of Boundary Integral Equations

2. Review of classical Boundary Element Methods

3. Principle of the Fast Multipole Method
   - Principle for the interaction between 2 cells
   - Basic ideas of the Multi-level algorithm
   - Other factorizations

4. Fast Multipole accelerated BEM

5. Algebraic fast BEMs based on low rank approximations
Principle for Coulomb potential

Computing pairwise interactions: gravitational potential in astrophysics

\[ G(x, y) = \frac{1}{|x - y|} \]

\[ u(x) = \sum_{y \in Y} G(x, y) f(y) \]

If A & B are 2 disjoint squares of the same size, containing \( O(N) \) points

- Direct: potential induced at points in A by points in B: \( O(N^2) \)
- Approximate calculation if A and B are well-separated

Approximate calculation between two cells

If A and B are well-separated: **Factorization** with cost $O(N)$

1. Sum up mass in B: $f_B = \sum_{y \in B} f(y)$; place it at $c_B$.
2. Evaluate $u_A = G(c_A; c_B) f_B$ at $c_A$ (i.e. all the mass is at $c_B$).
3. $u_A$: approximation of the potential at each point $x \in A$.

---

Outline

1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
   - Principle for the interaction between 2 cells
   - Basic ideas of the Multi-level algorithm
   - Other factorizations
4. Fast Multipole accelerated BEM
5. Algebraic fast BEMs based on low rank approximations
Back to pairwise interactions (multi-level)

Previous algorithm gives a partial answer to the initial problem. How do we compute all the interactions?

Partition of the domain with a quadtree (or octree in 3D)
- each cell has four children (or eight in 3D);
- $O(\log N)$ levels;
- at level $\ell$: $4^\ell$ cells;
- $O(N/4^\ell)$ points per cell.

Definitions
- Near field $N(B)$: union of $B$ and its neighbors;
- Far field $F(B)$: complement of $N(B)$;
- $b$: parent cell of $B$;
B at level 2: $\forall A \in F(B)$: interaction from B to A with \textbf{Factorization};

- Cost at this level:
- $\forall A \in N(B)$: we cannot process on this level, go down in the tree.
B at level 2: $\forall A \in F(B)$: interaction from B to A with **Factorization**;
- Cost at this level: $4^2 \cdot O(1) \cdot O(N/4^2) = O(N)$;
- $\forall A \in N(B)$: we cannot process on this level, go down in the tree.
Back to pairwise interactions (multi-level algorithm)

B at level 2: \( \forall A \in F(B) \): interaction from B to A with **Factorization**;
- Cost at this level: \( 4^2 \cdot O(1) \cdot O(N/4^2) = O(N) \);
- \( \forall A \in N(B) \): we cannot process on this level, go down in the tree.

B at level 3:
- Interactions with \( F(b) \) have already been computed at level 2;
- Remaining: interaction between \( B \) and \( N(b) \) (9 \( \times \) 4 cells in \( N(b) \));
- **Factorization** for the \( 36 - 9 = 27 \) well-separated cells;
- Cost at this level:
- \( \forall A \in N(B) \): go down in tree.
Back to pairwise interactions (multi-level algorithm)

B at level 2: $\forall A \in F(B)$: interaction from B to A with **Factorization**;
- Cost at this level: $4^2 \cdot O(1) \cdot O(N/4^2) = O(N)$;
- $\forall A \in N(B)$: we cannot process on this level, go down in the tree.

B at level 3:
- Interactions with $F(b)$ have already been computed at level 2;
- Remaining: interaction between $B$ and $N(b)$ ($9 \times 4$ cells in $N(b)$);
- **Factorization** for the $36 - 9 = 27$ well-separated cells;
- Cost at this level: $4^3 \cdot O(1) \cdot O(N/4^3) = O(N)$;
- $\forall A \in N(B)$: go down in tree.
Back to pairwise interactions (multi-level algorithm)

B at level 2: \( \forall A \in F(B) \): interaction from B to A with **Factorization**;

- Cost at this level: \( 4^2 \cdot O(1) \cdot O(N/4^2) = O(N) \);
- \( \forall A \in N(B) \): we cannot process on this level, go down in the tree.

B at level 3: **Factorization** for the 27 well-separated cells;

- Cost at this level: \( 4^3 \cdot O(1) \cdot O(N/4^3) = O(N) \);
Back to pairwise interactions (multi-level algorithm)

General $\ell$-level

- $4^\ell$ possible $B$; **Factorization** for the 27 *well-separated* cells;
- Cost of far field computation:
Back to pairwise interactions (multi-level algorithm)

General $\ell$-level

- $4^\ell$ possible $B$; **Factorization** for the 27 *well-separated* cells;
- Cost of far field computation: $4^\ell \cdot O(1) \cdot O(N/4^\ell) = O(N)$.
Back to pairwise interactions (multi-level algorithm)

General $\ell$-level
- $4^\ell$ possible $B$; **Factorization** for the 27 *well-separated* cells;
- Cost of far field computation:\[4^\ell \cdot O(1) \cdot O(N/4^\ell) = O(N)\];

Leaf level: direct computation $\forall A \in N(B)$:
Back to pairwise interactions (multi-level algorithm)

General $\ell$-level
- $4^\ell$ possible B; **Factorization** for the 27 *well-separated* cells;
- Cost of far field computation: $4^\ell \cdot O(1) \cdot O(N/4^\ell) = O(N)$;

Leaf level: direct computation $\forall A \in N(B)$: $O(N) \cdot O(1) \cdot O(1) = O(N)$. 

$\ell$-level

Leaf level
Total cost of the algorithm

- The cost at each level: \( O(N) \) → Algorithm is \( O(N \log N) \)
- \( O(\log N) \) levels

Improving the **Factorization**: first (last) step only depends on B (A)

1. For each level \( \ell \) and each cell A on level \( \ell \), set \( u_A = 0 \);
2. For each level \( \ell \) and each cell B on level \( \ell \),
   \[
   f_B = \sum_{y \in B \cap P} f(y);
   \]
3. For each level \( \ell \) and each cell B on level \( \ell \), and for each cell A in B’s interaction list, update
   \[
   u_A := u_A + G(c_A; c_B)f_B;
   \]
4. For each level \( \ell \) and each cell A on this level, update
   \[
   u(x) := u(x) + u_A \text{ for each } x \in A \cap P;
   \]
5. For each cell B on the leaf level, update
   \[
   u(x) := u(x) + \sum_{y \in N(B) \cap P} G(x; y)f(y).
   \]
Outline

1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
   - Principle for the interaction between 2 cells
   - Basic ideas of the Multi-level algorithm
   - Other factorizations
4. Fast Multipole accelerated BEM
5. Algebraic fast BEMs based on low rank approximations
Improving the multi-level algorithm

How to reduce the number of steps?

- $B_1, B_2, B_3, B_4$ children of $B$; $A_1, A_2, A_3, A_4$ children of $A$;

Suppose that we visit the parent cell after the children cells:

$$f_B = f_{B_1} + f_{B_2} + f_{B_3} + f_{B_4}$$

instead of summing over all $f(y)$ in B.

Suppose that we visit the children cells after the parent cell:

- Instead of updating $u(x) := u(x) + u_A$ for each $x \in A$
- Instead of updating $u_{A_i} := u_{A_i} + u_A$. 
Optimal multi-level algorithm

Initialization. For each level $\ell$ and each box $A$ on level $\ell$:

$$u_A = 0.$$

Upward pass. For level $L-1$ to level 0 and for each box $B$
- if $B$ is a leaf box: $f_B = \sum_{y \in B \cap P} f(y)$;
- If $B$ is not a leaf box: $f_B = f_{B_1} + f_{B_2} + f_{B_3} + f_{B_4}$.

Transfer. For each level $\ell$, for each $B$, and for each $A$ in $B$'s interaction list:

$$u_A := u_A + G(c_A; c_B)f_B$$

Downward pass. For level 0 to level $L-1$ and for each box $A$
- if $A$ is not a leaf box: $u_{A_i} := u_{A_i} + u_A$ for each child $A_i$ of $A$;
- If $A$ is a leaf box: $u(x) := u(x) + u_A$.

Near contributions. For each box $B$ on the leaf level:

$$u(x) := u(x) + \sum_{y \in N(B) \cap P} G(x; y)f(y)$$
Nested property of the far field

- $f_B$: approximation of the charge distribution inside $B$;
- Valid for any cell $A \in F(B) \rightarrow$ independent from $A$;
- Optimized evaluation of $f_B$ from the children cells only if $F(B) \subset F(B_i)$ for each $B_i$ (always true).

Cost of the upward and downward passes:

- $O(N)$ cells in the tree
- $O(1)$ per box

$\Rightarrow$ The cost of the complete algorithm is reduced to $O(N)$
Low-rank approximations

Accuracy

- With $f_B$ and $u_A$: the accuracy will be terrible since A and B are only one box away from each other;
- Poor rank-1 approximation of the interaction between A and B.

Better low-rank approximations

- Classical FMM: based on analytic properties of the kernel (i.e., complex analysis in 2D and spherical harmonics in 3D).
  - $f_B$ and $u_A$: multipole and local expansions.
- Kernel-independent FMM: reproduce the potential in A induced by points in B by placing some equivalent charges either in B or on its boundary.
  - $f_B$ and $u_A$: equivalent charges and check potentials.
Outline

1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
4. Fast Multipole accelerated BEM
5. Algebraic fast BEMs based on low rank approximations
Accelerating the BEM: oscillatory kernels

**EFIE:**

- Helmholtz fundamental solution: \( G(x - y) = \frac{\exp(ik|x - y|)}{4\pi|x - y|} \)
- Boundary integral equation to solve
  \[
  \int_{\Gamma} G(x - y)p(y)dS_y = -u^{inc}(x), \quad x \in \Gamma
  \]

**Linear system**

- \( \mathbf{A} \mathbf{p} = -\mathbf{u}^{inc} \) with \( \mathbf{A} \) fully-populated
- Accelerating the matrix-vector product
- Specificities due to the oscillatory nature of the kernel
Various FMMs for Helmholtz

Method based on multipole expansion: expensive at mid-frequencies


Method based on plane wave expansion: instabilities at low frequency


Combination of low- and mid-frequency FM techniques


⇒ We consider only the plane wave based FMM for limited to mid-frequencies
1. Principle of Boundary Integral Equations

2. Review of classical Boundary Element Methods

3. Principle of the Fast Multipole Method

4. Fast Multipole accelerated BEM
   - Plane wave expansion for mid-frequencies
     - Single-level algorithm
     - Multi-level algorithm
     - Implementation of the FMM: Numerical issues

5. Algebraic fast BEMs based on low rank approximations
Legendre Polynomials

Legendre Polynomials: $\ell \in \mathbb{N}$

\[
\frac{d}{dx}[(1 - x^2) \frac{d}{dx} P] + \ell (\ell + 1) P = 0; \quad P_0(x) = 1; P_1(x) = x
\]

- Orthonormality: \[
\int_{-1}^{1} P_{\ell}(x) P_{\ell'}(x) dx = \frac{2}{2\ell + 1} \delta_{\ell\ell'}
\]
- Recurrence relation: \[
(\ell + 1) P_{\ell+1} - (2\ell + 1)xP_\ell + \ell P_{\ell-1} = 0
\]

Associated Legendre Polynomials

\[
\frac{d}{dx}[(1 - x^2) \frac{d}{dx} P] + [\ell (\ell + 1) - \frac{m^2}{1 - x^2}] P = 0;
\]

- $P_{\ell}^m(x) = (-1)^m (1 - x^2)^{m/2} \frac{d}{dx^m} P_{\ell}(x)$

- Orthonormality: \[
\int_{-1}^{1} P_{\ell}^m(x) P_{\ell'}^m(x) dx = \frac{2}{2\ell + 1} \frac{(\ell + m)!}{(\ell - m)!} \delta_{\ell\ell'}
\]
Spherical harmonics

\[ \theta \in [0, \pi] \text{ and } \phi \in [0, 2\pi] \]

- \[ Y_{\ell m}(\theta, \phi) = \sqrt{\frac{2\ell + 1}{4\pi}} \frac{(\ell - m)!}{(\ell + m)!} P_{\ell}^{m}(\cos \theta) e^{im\phi} \]

- \[ \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} Y_{\ell m}(\theta, \phi) Y_{\ell m}^{*}(\theta', \phi') = \delta_{\ell \ell'} \delta_{mm'} \]

- On the unit sphere, any square-integrable function is expanded

\[ f(\theta, \phi) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} A_{\ell m} Y_{\ell m}(\theta, \phi); \quad A_{\ell m} = \int_{S^2} Y_{\ell m}^{*}(\theta, \phi) f(\theta, \phi) d\hat{s} \]
Spherical Bessel and Hankel functions

- Helmholtz equation \((\nabla^2 + k^2) \psi(x) = 0\)
- Separation of variables: \(\psi(x) = \sum_{\ell,m} f_{\ell m} Y_{\ell m}(\theta, \phi)\)
- Radial functions are solution to:
  \[
  \left[ \frac{d^2}{dr^2} + \frac{2}{r} \frac{d}{dr} + k^2 - \frac{\ell(\ell + 1)}{r^2} \right] f_{\ell}(r) = 0.
  \]
  Two linearly independent solutions to this equation: \(j_n\) et \(y_n\)
- Spherical Hankel functions \(h_n^{(1)}(x) = j_n(x) + iy_n(x)\)
  \[
  \frac{2\ell + 1}{x} z_{\ell}(x) = z_{\ell-1}(x) + z_{\ell+1}(x)
  \]
Decomposition of Helmholtz fundamental sol.

Approximation of the kernel: separation of the variables

\[
\begin{align*}
    r &= x - y = (y_0 - y) + (x_0 - y_0) - (x_0 - x) \\
    \frac{\exp(i k |x - y|)}{4\pi |x - y|} &= \lim_{L \to +\infty} \int_{\hat{s} \in S^2} e^{i k \hat{s} \cdot (y - y_0)} G_L(\hat{s}; r_0; k) e^{-i k \hat{s} \cdot (x - x_0)} d\hat{s},
\end{align*}
\]
Decomposition of Helmholtz fundamental sol.

Approximation of the kernel: separation of the variables

\[ r = x - y = (y_0 - y) + (x_0 - y_0) - (x_0 - x) \]

\[ \exp \left( \frac{ik |x - y|}{4\pi |x - y|} \right) = \lim_{L \to +\infty} \int_{\hat{s} \in S^2} e^{ik \hat{s}.(y - y_0)} G_L(\hat{s}; r_0; k) e^{-ik \hat{s}.(x - x_0)} d\hat{s}, \]

Transfer function

\[ G_L(\hat{s}; r_0; k) = \frac{ik}{16\pi^2} \sum_{p=0}^{L} (2p + 1)i^p h_p^{(1)}(k|r_0|) P_p(\cos(\hat{s}, r_0)) \]
Remark: Extension to other kernels

Extension of FMM for Helmholtz to elastodynamics

\[ U_i^k(x, y; \omega) = \frac{1}{k_S^2 \mu} \left( (\delta_{qs} \delta_{ik} - \delta_{qk} \delta_{is}) \frac{\partial}{\partial x_q} \frac{\partial}{\partial y_s} G_S(|x - y|) + \frac{\partial}{\partial x_i} \frac{\partial}{\partial y_k} G_P(|x - y|) \right) \]

\[ T_i^k(x, y; \omega) = C_{ijh\ell} \frac{\partial}{\partial y_\ell} U_h^k(x, y; \omega) n_j(y) \]

\[ G_\alpha(z) = \frac{\exp(i k_\alpha z)}{4\pi z} \quad \text{(fund. sol. Helmholtz eqn., } \alpha = P, S) \]

Decomposition of elastodynamics fundamental solution

\[ U_i^k(x, y; \omega) = \lim_{L \to +\infty} \int_{\hat{s} \in S} e^{i k_P \hat{s} \cdot (y - y_0)} U_{i, L}^{k, P} (\hat{s}; r_0) e^{-i k_P \hat{s} \cdot (x - x_0)} d\hat{s} \]

\[ + \lim_{L \to +\infty} \int_{\hat{s} \in S} e^{i k_S \hat{s} \cdot (y - y_0)} U_{i, L}^{k, S} (\hat{s}; r_0) e^{-i k_S \hat{s} \cdot (x - x_0)} d\hat{s} \]

where

\[ U_{i, L}^{k, S} (\hat{s}; r_0) = \frac{1}{\mu} (\delta_{ik} - \hat{s}_k \hat{s}_i) G_L(\hat{s}; r_0; k_S), \]

\[ U_{i, L}^{k, P} (\hat{s}; r_0) = \frac{\gamma^2 \hat{s}_i \hat{s}_k}{{\mu}} G_L(\hat{s}; r_0; k_P) \quad \text{with} \quad \gamma = k_P / k_S \]

- Multipole expansion of same form for \( T_i^k \)
Outline

1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
4. Fast Multipole accelerated BEM
   - Plane wave expansion for mid-frequencies
   - Single-level algorithm
   - Multi-level algorithm
   - Implementation of the FMM: Numerical issues
5. Algebraic fast BEMs based on low rank approximations
Theorem

\[
\frac{\exp(ik|\mathbf{r} + \mathbf{r}_0|)}{|\mathbf{r} + \mathbf{r}_0|} = \lim_{L \to +\infty} \int_{\hat{s} \in S^2} e^{ik\hat{s} \cdot \mathbf{r}} G_L(\hat{s}; \mathbf{r}_0) d\hat{s}, \quad \forall \mathbf{r}_0 \in \mathbb{R}^3.
\]

\[\forall \mathbf{r} \in \mathbb{R}^3 \text{ such that } |\mathbf{r}| < |\mathbf{r}_0|.
\]

Convergence of expansion assured if \(x\) and \(y\) lie in non-adjacent cells
Theorem

\[ \frac{\exp(ik|\mathbf{r} + \mathbf{r}_0|)}{|\mathbf{r} + \mathbf{r}_0|} = \lim_{L \to +\infty} \int_{\hat{s} \in S^2} e^{ik \hat{s}.\mathbf{r}} G_L(\hat{s}; \mathbf{r}_0) d\hat{s}, \quad \forall \mathbf{r}_0 \in \mathbb{R}^3. \]

\[ \forall \mathbf{r} \in \mathbb{R}^3 \text{ such that } |\mathbf{r}| < |\mathbf{r}_0|. \]

Convergence of expansion assured if \( x \) and \( y \) lie in non-adjacent cells

\( \Rightarrow \) System matrix split into near and far parts
Algorithm for the single-level FMM

Integration cell

\[ C_y \]

Collocation cell

\[ C_x \]

\[ \int_{\Gamma} G(\mathbf{x} - \mathbf{y}) p(\mathbf{y}) dS_y \]
Algorithm for the single-level FMM

Integration cell

\[ y_1 \quad y_2 \quad y_3 \quad y_4 \]
\[ y_0 \]

Collocation cell

\[ x_1 \quad x_3 \]
\[ x_0 \quad x_2 \quad x_4 \]

\[ C_y \]
\[ C_x \]

1. Compute multipole moments for each \( C_y \) and \( \hat{s} \in S^2 \)

\[ \mathcal{R}(\hat{s}; C_y) = \int_{\partial\Omega \cap C_y} p(y) e^{ik\hat{s} \cdot (y-y_0)} dS_y \]
Algorithm for the single-level FMM

1. Compute multipole moments for each $C_y$ and $\hat{s} \in S^2$

$$R(\hat{s}; C_y) = \int_{\partial \Omega \cap C_y} p(y) e^{ik\hat{s} \cdot (y - y_0)} dS_y$$

2. Transfer from $C_y$ to non-adjacent $C_x$

$$L(\hat{s}; C_x) = \sum_{C_y \notin A(C_x)} G_L(\hat{s}; r_0; k) R(\hat{s}; C_y)$$

$$\int_{\Gamma} G(x - y)p(y)dS_y$$
Algorithm for the single-level FMM

1. Compute multipole moments for each $C_y$ and $\hat{s} \in S^2$

\[
\mathcal{R}(\hat{s}; C_y) = \int_{\partial\Omega \cap C_y} p(y) e^{i k \hat{s} \cdot (y - y_0)} dS_y
\]

2. Transfer from $C_y$ to non-adjacent $C_x$

\[
\mathcal{L}(\hat{s}; C_x) = \sum_{C_y \notin A(C_x)} G_L(\hat{s}; r_0; k) \mathcal{R}(\hat{s}; C_y)
\]

3. Evaluate FM contribution $(\mathcal{K}p)^{FM}(\mathbf{x})$ to matrix-vector product

\[
(\mathcal{K}p)^{FM}(\mathbf{x}) \approx \sum_q w_q e^{-i k \hat{s}_q \cdot (\mathbf{x} - \mathbf{x}_0)} \mathcal{L}(\hat{s}_q; C_x)
\]
Algorithm for the single-level FMM

1. Compute multipole moments for each $C_y$ and $\hat{s} \in S^2$

\[
\mathcal{R}(\hat{s}; C_y) = \int_{\partial \Omega \cap C_y} p(y) e^{ik\hat{s} \cdot (y-y_0)} dS_y
\]

2. Transfer from $C_y$ to non-adjacent $C_x$

\[
\mathcal{L}(\hat{s}; C_x) = \sum_{C_y \notin \mathcal{A}(C_x)} \mathcal{G}_L(\hat{s}; r_0; k) \mathcal{R}(\hat{s}; C_y)
\]

3. Evaluate FM contribution $(Kp)^{FM}(x)$ to matrix-vector product

\[
(Kp)^{FM}(x) \approx \sum_q w_q e^{-ik\hat{s}_q \cdot (x-x_0)} L(\hat{s}_q; C_x)
\]

4. Add near contribution $(Ku)^{near}(x)$ to matrix-vector product (computed using standard BEM techniques)
Single-region FMM (homogeneous domain)

Complexity of single-level FMM: $O(N^{3/2})$ per GMRES iteration instead of $O(N^2)$ per GMRES iteration (see TD)

![Diagram showing Standard BEM and FM-BEM](image)

- **Standard BEM**
- **FM-BEM**
Single-region FMM (homogeneous domain)

**Complexity of single-level FMM:** $O(N^{3/2})$ per GMRES iteration instead of $O(N^2)$ per GMRES iteration (see TD)

---

**Improve FMM computational efficiency:** multi-level FMM (next session)
Main approximations

\[
\frac{\exp(ik|x - y|)}{4\pi |x - y|} = \lim_{L \to +\infty} \int_{\hat{s} \in S^2} e^{ik\hat{s} \cdot (y - y_0)} G_L(\hat{s}; r_0) e^{-ik\hat{s} \cdot (x - x_0)} d\hat{s},
\]

\[
G_L(\hat{s}; r_0) = \frac{ik}{16\pi^2} \sum_{p=0}^{L} (2p + 1) i^p h_p^{(1)}(k|r_0|) P_p(\cos(\hat{s}, r_0))
\]

Discrete approximation of the kernel:

1. Choice of a finite \( L \);
2. Discretization of the unit sphere;

\[
\frac{\exp(ik|x - y|)}{4\pi |x - y|} \approx \sum_{p} \omega_p e^{ik\hat{s}_p \cdot (y - y_0)} G_L(\hat{s}_p; r_0) e^{-ik\hat{s}_p \cdot (x - x_0)}
\]
Outline

1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
4. Fast Multipole accelerated BEM
   - Plane wave expansion for mid-frequencies
   - Single-level algorithm
   - Multi-level algorithm
   - Implementation of the FMM: Numerical issues
5. Algebraic fast BEMs based on low rank approximations
Principle of the Multi-level FMM

level $\ell = 0$

level $\ell = 1$

level $\ell = 2$

level $\ell = 3$

... level $\ell = \bar{\ell}$ (leaf)

- computation organization based on recursive subdivision: octree
- non-FMM calculations confined to smallest spatial region;
- FMM calculations performed between large groups;
- octree: start at level $\ell = 2$

$\Rightarrow$ Complexity of multi-level FMM: $O(N \log N)$ instead of $O(N^2)$
Schematic representation of the multi-level FMM

Level $\bar{\ell}$

$R(\hat{s}; C_y^{(\bar{\ell})})$ \hspace{0.5cm} \rightarrow \hspace{0.5cm} L(\hat{s}; C_y^{(\bar{\ell})})$

Upward

Level $\bar{\ell} - 1$

$R(\hat{s}; C_y^{(\bar{\ell} - 1)})$ \hspace{0.5cm} \rightarrow \hspace{0.5cm} L(\hat{s}; C_y^{(\bar{\ell} - 1)})$

Upward

Level $\bar{\ell}$

$R(\hat{s}; C_y^{(\bar{\ell})})$ \hspace{0.5cm} \rightarrow \hspace{0.5cm} L(\hat{s}; C_y^{(\bar{\ell})})$

Integration

Transfer

Downward

...
Multi-level FM-BEM for EFIE

- Multipole moments evaluated for leaf cells

\[ \mathcal{R}(\hat{s}; C_y^{(\ell)}) = \int_{\partial \Omega \cap C_y} p(y) e^{ik\hat{s}.(y-y_0^{(\ell)})} dS_y \]

- Upward pass: multipole moments for level-\( \ell \) cells by aggregation of those for level-\((\ell + 1)\) (children) cells

- Transfers (multipole-to-local) at highest possible level

- Downward pass: local expansions for level-\( \ell \) cells evaluated from those for level-\((\ell - 1)\) (parent) cells

- Aggregation of FM contributions to \((\mathcal{K}_p)\) at leaf cell level

- Near contributions to \((\mathcal{K}_p)\) added at leaf cell level
Multi-level FM-BEM for EFIE

- Multipole moments evaluated for leaf cells

- **Upward pass**: multipole moments for level-$\ell$ cells by aggregation of those for level-$(\ell + 1)$ (children) cells

  \[
  R(\hat{s}; C_y^{(\ell)}) = \sum_{C_{y+1}^{(\ell+1)} \in S(C_y^{(\ell)})} e^{-ik\hat{s} \cdot (y_{0+1}^{(\ell+1)} - y_0^{(\ell)})} R(\hat{s}; C_{y+1}^{(\ell+1)})
  \]

- Transfers (multipole-to-local) at highest possible level

- **Downward pass**: local expansions for level-$\ell$ cells evaluated from those for level-$(\ell - 1)$ (parent) cells

- Aggregation of FM contributions to $(K_p)$ at leaf cell level

- Near contributions to $(K_p)$ added at leaf cell level
Multi-level FM-BEM for EFIE

- Multipole moments evaluated for leaf cells

- **Upward pass:** multipole moments for level-$\ell$ cells by aggregation of those for level-$(\ell + 1)$ (children) cells

- Transfers (multipole-to-local) at highest possible level
  \[
  \mathcal{L}(\mathbf{s}; C_x^{(\ell)}) = \sum_{C_y^{(\ell)} \in \mathcal{I}(C_x^{(\ell)})} \mathcal{G}_L(\mathbf{s}; r_0) \mathcal{R}(\mathbf{s}; C_y^{(\ell)})
  \]

- **Downward pass:** local expansions for level-$\ell$ cells evaluated from those for level-$(\ell - 1)$ (parent) cells

- Aggregation of FM contributions to $(\mathcal{K}_p)$ at leaf cell level

- Near contributions to $(\mathcal{K}_p)$ added at leaf cell level
Multi-level FM-BEM for EFIE

- Multipole moments evaluated for leaf cells

- **Upward pass**: multipole moments for level-$\ell$ cells by aggregation of those for level-$(\ell + 1)$ (children) cells

- Transfers (multipole-to-local) at highest possible level

- **Downward pass**: local expansions for level-$\ell$ cells evaluated from those for level-$(\ell - 1)$ (parent) cells

\[
\mathcal{L}(\hat{s}; C^{(\ell)}_x) = \mathcal{L}(\hat{s}; C^{(\ell)}_x) + e^{-ik(\hat{s} \cdot (x_0^{(\ell-1)} - x_0^{(\ell)}))} \mathcal{L}(\hat{s}; C^{(\ell-1)}_x)
\]

- Aggregation of FM contributions to $(\mathcal{K}_p)$ at leaf cell level

- Near contributions to $(\mathcal{K}_p)$ added at leaf cell level
Multi-level FM-BEM for EFIE

- Multipole moments evaluated for leaf cells

- **Upward pass**: multipole moments for level-$\ell$ cells by aggregation of those for level-$(\ell + 1)$ (children) cells

- Transfers (multipole-to-local) at highest possible level

- **Downward pass**: local expansions for level-$\ell$ cells evaluated from those for level-$(\ell - 1)$ (parent) cells

- Aggregation of FM contributions to $(\mathcal{K}_p)$ at leaf cell level
  \[
  (\mathcal{K}_p)^{FM}(x) \approx \sum_q \omega_q e^{-ik \hat{s}_q \cdot (x - x_0^{(\ell)})} \mathcal{L}(\hat{s}_q; C_x)
  \]

- Near contributions to $(\mathcal{K}_p)$ added at leaf cell level
Outline

1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
4. Fast Multipole accelerated BEM
   - Plane wave expansion for mid-frequencies
   - Single-level algorithm
   - Multi-level algorithm
   - Implementation of the FMM: Numerical issues
5. Algebraic fast BEMs based on low rank approximations
Main approximations

\[
\frac{\exp(ik|x - y|)}{4\pi|x - y|} = \lim_{L \to +\infty} \int_{\hat{s} \in S^2} e^{ik\hat{s} \cdot (y - y_0)} G_L(\hat{s}; r_0) e^{-ik\hat{s} \cdot (x - x_0)} d\hat{s},
\]

\[
G_L(\hat{s}; r_0) = \frac{ik}{16\pi^2} \sum_{p=0}^{L} (2p + 1)i^p h_p^{(1)}(k|r_0|) P_p(\cos(\hat{s}, r_0))
\]

Discrete approximation of the kernel:

1. Choice of a finite \( L \);
2. Discretization of the unit sphere;

\[
\frac{\exp(ik|x - y|)}{4\pi|x - y|} \approx \sum_p \omega_p e^{ik\hat{s}_p \cdot (y - y_0)} G_L(\hat{s}_p; r_0) e^{-ik\hat{s}_p \cdot (x - x_0)}
\]
Truncation of the transfer function

\[ h_0(k|r_0 + r|) = \frac{1}{ik} \lim_{L \to +\infty} \int_{S^2} e^{ik\hat{s} \cdot r} g_L(\hat{s}; r_0) d\hat{s} \]

\[ \approx \int_{S^2} e^{ik\hat{s} \cdot r} \sum_{p=0}^{L} \frac{(2p + 1)}{4\pi} i^p h_p^{(1)}(k|r_0|) P_p(\cos(\hat{s}, r_0)) \]

\[ \approx \sum_{p=0}^{L} (-1)^p (2p + 1) h_p^{(1)}(k|r_0|) j_p(k|r|) P_p(\cos(r_0, r)) \]

- \text{\(L\) too small: convergence not reached for plane wave expansion; }
- \text{\(L\) can be chosen independently from \(r_0\);}
- \text{\(L\) too large: divergence of \(h_p^{(1)}\) (numerical instabilities);}

Error estimate: \(\exists C_1, C_2\) such that, for any \(\epsilon < 1\), \(L = C_1 k|r| + C_2 \ln(k|r|)\)

\[ \Rightarrow \left| h_0(k|r_0 + r|) - \sum_{p=0}^{L} (-1)^p (2p+1) h_p^{(1)}(k|r_0|) j_p(k|r|) P_p(\cos(r_0, r)) \right| \leq \epsilon \]

Truncation of the transfer function

\[ h_0(k|\mathbf{r}_0 + \mathbf{r}|) = \frac{1}{i k} \lim_{L \to +\infty} \int_{S^2} e^{i k \mathbf{s} \cdot \mathbf{r}} G_L(\mathbf{s}; \mathbf{r}_0) d\mathbf{s} \]

\[ \approx \int_{S^2} e^{i k \mathbf{s} \cdot \mathbf{r}} \sum_{p=0}^{L} \frac{(2p + 1)}{4\pi} i^p h_p^{(1)}(k|\mathbf{r}_0|) P_p(\cos(\mathbf{s}, \mathbf{r}_0)) \]

\[ \approx \sum_{p=0}^{L} (-1)^p (2p + 1) h_p^{(1)}(k|\mathbf{r}_0|) j_p(k|\mathbf{r}|) P_p(\cos(\mathbf{r}_0, \mathbf{r})) \]

- \( L \) too small: convergence not reached for plane wave expansion;
- \( L \) can be chosen independently from \( \mathbf{r}_0 \);
- \( L \) too large: divergence of \( h_p^{(1)} \) (numerical instabilities);

Error estimate: \( \exists C_1, C_2 \) such that, for any \( \epsilon < 1 \), \( L = C_1 k|\mathbf{r}| + C_2 \ln(k|\mathbf{r}|) \)

\[ \Rightarrow \left| h_0(k|\mathbf{r}_0 + \mathbf{r}|) - \sum_{p=0}^{L} (-1)^p (2p + 1) h_p^{(1)}(k|\mathbf{r}_0|) j_p(k|\mathbf{r}|) P_p(\cos(\mathbf{r}_0, \mathbf{r})) \right| \leq \epsilon \]


This formulation is not accurate at low frequencies due to \( h_p^{(1)} \)
Truncation of transfer function: practical aspect

\[ L = \sqrt{3}k_Sd + C_\varepsilon \log_{10}(\sqrt{3}k_Sd + \pi) \]

\[ \Rightarrow C_\varepsilon = 7.5 \]

Numerical integration on the unit sphere

\[
\int_{\hat{s} \in S^2} e^{ik \hat{s} . (y - y_0)} G_L(\hat{s}; r_0) e^{-ik \hat{s} . (x - x_0)} d\hat{s} \approx \sum_p \omega_p e^{ik \hat{s}_p . (y - y_0)} G_L(\hat{s}_p; r_0) e^{-ik \hat{s}_p . (x - x_0)}
\]

General idea: Approximation by spherical harmonics

- Expansion on a basis of spherical harmonics

\[
e^{ik \hat{s} . (y - y_0)} G_L(\hat{s}; r_0) e^{-ik \hat{s} . (x - x_0)} \approx \sum_{0 \leq \ell \leq L \atop -\ell \leq m \leq \ell} A_{\ell m} Y_{\ell m}(\hat{s})
\]

There are error estimates to quantify this approximation

- Product rule in angular spherical coordinates \((\phi, \theta)\):
  - \(\phi\): \(2L + 1\)-point uniform rule over \([0, 2\pi]\)
  - \(\theta\): \(L + 1\)-point Gauss-Legendre rule over \([0, \pi]\)
  - \(Q = (L + 1)(2L + 1)\) quadrature points overall
\[
\frac{\exp(ik|x - y|)}{|x - y|} = \lim_{L \to +\infty} \int_{\hat{s} \in S} e^{ik\hat{s} \cdot (y_0 - y)} G_L(\hat{s}; r_0; k) e^{-ik\hat{s} \cdot (x - x_0)} d\hat{s}
\]

- Truncation parameter \( L \) chosen according to empirical formula
  \[
  L = \sqrt{3}kd + C_\epsilon \log_{10}(\sqrt{3}kd + \pi)
  \]

- \( \ell \) determined such that \( d^{(\ell)} \approx 0.3\lambda \) for smallest ("leaf") cell
  - influence on CPU
  - influence on accuracy

- Numerical quadrature over \( S \): \( Q = O(L^2) \) quadrature points
  - \( Q \) is level-dependent (grows approx. quadratically with cell size)

- Upward (downward) passes: direct (inverse) extrapolation step

**Complexity analysis:** \( O(N \log N) \) per iter. (instead of \( O(N^2) \) per iter.)
Advantages/Issues of the FMM

Advantages

- Solid mathematical background
- Controlled Accuracy
- Theoretical Complexity estimate

Issues

- Kernel dependent method: change of kernel implies major modifications
- Another method for low-frequency regime (add evanescent waves to propagative waves)
- Wideband FMM: Complex algorithm to deal with high and low frequencies

⇒ Are there alternative approaches to speed up the solution?
Outline

1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
4. Fast Multipole accelerated BEM
5. Algebraic fast BEMs based on low rank approximations
Outline

1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
4. Fast Multipole accelerated BEM
5. Algebraic fast BEMs based on low rank approximations
   - Principle of Low rank approximations
   - Determining low-rank approximations
   - $\mathcal{H}$-matrices based Solvers
Algebraic separation of the variables

\[
\int_{\Gamma} G(x - y) p(y) dS_y = -u^{inc}(x), \quad x \in \Gamma
\]

Structure of the BEM matrix: \( A p = b \)

- **RHS:** \( b_i = -u^{inc}(x^i) \)
- \( N \) collocation and interpolation points
- Interpolation of the unknown \( p(y) \): \( \phi \)
- Integration on the reference triangle with \( N_g \) Gauss points

\[
\begin{bmatrix}
  b_1 \\
  \vdots \\
  b_N \\
\end{bmatrix} = \begin{bmatrix}
  G(x_c, y_g) \\
  \vdots \\
  w_{N_g} \\
\end{bmatrix} \begin{bmatrix}
  w_1 \\
  \vdots \\
  w_{N_g} \\
\end{bmatrix} \begin{bmatrix}
  \phi_1(y_1) & \cdots & \phi_N(y_1) \\
  \vdots & \ddots & \vdots \\
  \phi_1(y_{N_g}) & \cdots & \phi_N(y_{N_g}) \\
\end{bmatrix} \begin{bmatrix}
  p_1 \\
  \vdots \\
  p_N \\
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
  b_1 \\
  \vdots \\
  b_N \\
\end{bmatrix}
\]
Algebraic separation of the variables

\[ \int_{\Gamma} G(\mathbf{x} - \mathbf{y}) p(\mathbf{y}) dS_y = -u^{inc}(\mathbf{x}), \quad \mathbf{x} \in \Gamma \]

Structure of the BEM matrix: \( \mathbf{A} \mathbf{p} = \mathbf{b} \)

- RHS: \( b_i = -u^{inc}(x^i) \)
- \( N \) collocation and interpolation points
- Interpolation of the unknown \( p(\mathbf{y}) \): \( \phi \)
- Integration on the reference triangle with \( N_g \) Gauss points

\[
\begin{bmatrix}
    b_1 \\
    \vdots \\
    b_N
\end{bmatrix} =
\begin{bmatrix}
    G(x_c, y_g)
\end{bmatrix}
\begin{bmatrix}
    \phi_1(y_1)w_1 & \cdots & \phi_N(y_1)w_1 \\
    \vdots & & \vdots \\
    \phi_1(y_{N_g})w_{N_g} & \cdots & \phi_N(y_{N_g})w_{N_g}
\end{bmatrix}
\begin{bmatrix}
    p_1 \\
    \vdots \\
    p_N
\end{bmatrix}
\]
Algebraic separation of the variables

\[ \int_{\Gamma} G(x - y)p(y)\,dS_y = -u^{inc}(x), \quad x \in \Gamma \]

Structure of the BEM matrix: \( A p = b \)

- **RHS:** \( b_i = -u^{inc}(x^i) \)
- \( N \) collocation and interpolation points
- Interpolation of the unknown \( p(y) \): \( \phi \)
- Integration on the reference triangle with \( N_g \) Gauss points

\[
\begin{bmatrix}
    b_1 \\
    \vdots \\
    b_N
\end{bmatrix} =
\begin{bmatrix}
    G(x_c, y_g)
\end{bmatrix}
\begin{bmatrix}
    \phi_1(y_1)w_1 & \ldots & \phi_N(y_1)w_1 \\
    \vdots & & \vdots \\
    \phi_1(y_{N_g})w_{N_g} & \ldots & \phi_N(y_{N_g})w_{N_g}
\end{bmatrix}
\begin{bmatrix}
    p_1 \\
    \vdots \\
    p_N
\end{bmatrix}
\]

\( A \)

⇒ We want to separate the variables \( x_c \) and \( y_g \)
Algebraic fast BEM

Best-possible factorization: rank one approximation

$$\int_{\Gamma} G(x - y) p(y) dS_y \approx u(x) \int_{\Gamma} v(y) p(y) dS_y$$

What can we expect in practice?

- $U$ collects the information for the collocation points
- $V$ collects the information for the interpolation points
- The idea is to perform a low rank approximation of the matrix

$$A \approx UV^T$$

- $U$ and $V$ matrices with $k$ columns

Advantages of this algebraic approach?

- Factorization of the operations: matrix-vector product

$$Ap = U(V^T p)$$

- No need of an analytic expression (algebraic)
- Possibility to combine with a direct solver
Outline

1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
4. Fast Multipole accelerated BEM
5. Algebraic fast BEMs based on low rank approximations
   - Principle of Low rank approximations
   - Determining low-rank approximations
   - $\mathcal{H}$-matrices based Solvers
Singular Value Decomposition

Rank of a matrix

- Column rank: max # of linearly independent column vectors
- Column and row ranks are equal $\Rightarrow$ rank of the matrix

Definition (Singular Value Decomposition)

$M \in \mathbb{C}^{m \times n}$ with $\text{rank}(M) = r$. The Singular Value Decomposition (SVD) of $M$ is to

- choose an orthogonal basis $v_1, \ldots, v_r$ of row space of $M$, and
- choose an orthogonal basis $u_1, \ldots, u_r$ of column space of $M$
- so that $Mv_i = \sigma_i u_i$, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$. 

$92 / 141$ S. Chaillat-Loseille  Fast algorithms for solving scattering pb by BEM
Theorem (Singular Value Decomposition)

\( M \in \mathbb{C}^{m \times n} \), there exists a factorization of \( M \) of the form \( M = U \Sigma V^* \)

- \( U \) and \( V \) are unitary matrices: \( U^T U = I_m \) and \( V^T V = I_n \)
- \( \Sigma \) is a diagonal matrix (singular values)

\( \Rightarrow \) The storage is reduced to \( O(mr + r + nr) \)

What is the link between the rank and a low-rank approximation?

So far, the SVD does not give an approximation but only a factorization.

Definition (Truncated SVD)

\( \mathbf{M}_r \) is the SVD of \( \mathbf{M} \) truncated to the \( r \) largest singular values

\[
\mathbf{M}_r = \sum_{i=1}^{r} \mathbf{U}_i \Sigma_{ii} \mathbf{V}_i^*. 
\]

Definition (Numerical rank)

\[
 k(\varepsilon) := \min \{ r \mid \| \mathbf{M} - \mathbf{M}_r \| \leq \varepsilon \| \mathbf{M} \| \} 
\]

⇒ We use a truncated SVD to determine a low-rank approximation.
SVD and low-rank approximations

What is the link between the rank and a low-rank approximation?
So far, the SVD does not give an approximation but only a factorization.

Definition (Truncated SVD)

\( \mathbf{M}_r \) is the SVD of \( \mathbf{M} \) truncated to the \( r \) largest singular values

\[
\mathbf{M}_r = \sum_{i=1}^{r} U_i \Sigma_{ii} V_i^*. 
\]

Definition (Numerical rank)

\[
k(\varepsilon) := \min \{ r \mid \| \mathbf{M} - \mathbf{M}_r \| \leq \varepsilon \| \mathbf{M} \| \} \]

\( \Rightarrow \) We use a truncated SVD to determine a low-rank approximation.

Unitary invariant norm: \( \| \mathbf{UMV} \| = \| \mathbf{M} \| \) for all unitary matrices \( \mathbf{U} \) and \( \mathbf{V} \)

- Spectral or 2-norm: \( \| \mathbf{M} \|_2 = \sigma_1 \) (\( \sigma_1 \) largest singular value)
- Frobenius norm: \( \| \mathbf{M} \|_F^2 = \sum_{i,j} |M_{ij}|^2 \)
Theorem (Eckart-Young, Best low rank approximation)

$M \in \mathbb{C}^{m \times n}$ with $m \geq n$, and $\| \cdot \|$ a unitary invariant norm. The best rank-$k$ approximation $M_k$ of $M$ defined such that

$$M_k := \min \left\{ \| M - R \| \mid R \in \mathbb{C}^{m \times n}, \text{rank}(R) \leq k \right\} = \| M - M_k \|$$

is $M_k = \sum_{i=1}^{k} U_i \Sigma_i V_i^*$, with $M = U \Sigma V^*$.

In addition, $\| M - M_k \|_F^2 = \sum_{i=k+1}^{n} \sigma_i^2$ and $\| M - M_k \|_2 = \sigma_{k+1}$.

⇒ Truncated SVD is the best low-rank approximation for $L^2$-norm.
Low-rank approximations: finding the main information

Representing concepts hidden in massive datasets: matrices are used to

- Evaluate the importance of Web pages (number of occurrences of a term is easy to fool, add the links between pages)
- Represent social networks
- Google, Facebook, Netflix, ...

Example: Finding concepts underlying movies

\[
\begin{bmatrix}
0.14 & 0 \\
0.42 & 0 \\
0.56 & 0 \\
0.70 & 0 \\
\end{bmatrix}
\begin{bmatrix}
12.4 & 0 & 0 \\
9.5 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 \\
0 & 0 & 0 & 0.71 \\
0 & 0 & 0 & 0.71 \\
\end{bmatrix}
\]

Low-rank approximations: finding the main information

Image Compression: the goal is to reduce the storage
- Images represented as matrices of size \( n \) times \( m \) pixels
- Gray scale images: 1 number per pixel
- Color images: 3 numbers per pixel (red, green and blue)

⇒ SVD: form the best rank-k approximations for the matrix

⇒ Truncated SVD to remove the redundant information
Low-rank approximations: finding the main information

Image Compression: the goal is to reduce the storage

- Images represented as matrices of size $n \times m$ pixels
- Gray scale images: 1 number per pixel
- Color images: 3 numbers per pixel (red, green and blue)

$\Rightarrow$ SVD: form the best rank-$k$ approximations for the matrix

$k = 10 \quad k = 50 \quad k = 200 \quad k = 1024$

$\Rightarrow$ Truncated SVD to remove the redundant information
Application to the BEM

\[ \int_{\Gamma} G(\mathbf{x} - \mathbf{y}) p(\mathbf{y}) dS_y = -u^{inc}(\mathbf{x}), \quad \mathbf{x} \in \Gamma \]

\[
\begin{bmatrix}
  b_1 \\
  \vdots \\
  b_N
\end{bmatrix} = 
\begin{bmatrix}
  G(\mathbf{x}_c, \mathbf{y}_g) \\
  \vdots \\
  w_{Ng}
\end{bmatrix} 
\begin{bmatrix}
  w_1 \\
  \vdots \\
  w_{Ng}
\end{bmatrix} 
\begin{bmatrix}
  \phi_1(\mathbf{y}_1) & \cdots & \phi_N(\mathbf{y}_1) \\
  \vdots & \ddots & \vdots \\
  \phi_1(\mathbf{y}_{Ng}) & \cdots & \phi_N(\mathbf{y}_{Ng})
\end{bmatrix} 
\begin{bmatrix}
  p_1 \\
  \vdots \\
  p_N
\end{bmatrix}
\]

How to speed-up the solution of \( \mathbf{A} \mathbf{p} = \mathbf{b} \)?

- Compute the SVD of \( \mathbf{A} \)
- According to the prescribed error, determine the low-rank approximation with the truncated SVD (memory costs reduced)
- Use of iterative solver: matrix-vector product (CPU reduced)
Application to the BEM

\[
\int_{\Gamma} G(\mathbf{x} - \mathbf{y}) p(\mathbf{y}) dS_y = -u^{inc}(\mathbf{x}), \quad \mathbf{x} \in \Gamma
\]

\[
\begin{bmatrix}
    b_1 \\
    \vdots \\
    b_N
\end{bmatrix} = \begin{bmatrix}
    G(\mathbf{x}_c, \mathbf{y}_g) \\
    \vdots \\
    w_{N_g}
\end{bmatrix} \begin{bmatrix}
    w_1 \\
    \vdots \\
    w_{N_g}
\end{bmatrix} \begin{bmatrix}
    \phi_1(\mathbf{y}_1) & \cdots & \phi_N(\mathbf{y}_1) \\
    \vdots & \ddots & \vdots \\
    \phi_1(\mathbf{y}_{N_g}) & \cdots & \phi_N(\mathbf{y}_{N_g})
\end{bmatrix} \begin{bmatrix}
    p_1 \\
    \vdots \\
    p_N
\end{bmatrix}
\]

**How to speed-up the solution of** \( A p = b \)?

- Compute the SVD of \( A \)
- According to the prescribed error, determine the low-rank approximation with the truncated SVD (memory costs reduced)
- Use of iterative solver: matrix-vector product (CPU reduced)

⇒ Matrix fully-populated but truncated SVD removes redundant information
⇒ Such approximation is the algebraic counterpart of the FMM
Low-rank matrices

Representation of low-rank matrices: $\mathcal{R}k$-matrices

$$M = AB^T$$

- $A \in \mathbb{R}^{m \times k}$ and $B \in \mathbb{R}^{n \times k}$
- Storage: $k(m + n)$ (instead of $mn$)
- Link with the SVD: $A = U\Sigma$ and $B = V$

Definition (Low-rank matrices)

$M \in \mathbb{R}^{m \times n}$ of rank $k$ is called low-rank if

$$k(m + n) \ll mn$$

Matrix-vector multiplication: $Mx = y$
Low-rank matrices

Representation of low-rank matrices: $\mathcal{R}k$–matrices

$$M = AB^T$$

- $A \in \mathbb{R}^{m \times k}$ and $B \in \mathbb{R}^{n \times k}$
- Storage: $k(m + n)$ (instead of $mn$)
- Link with the SVD: $A = U\Sigma$ and $B = V$

Definition (Low-rank matrices)

$M \in \mathbb{R}^{m \times n}$ of rank $k$ is called low-rank if

$$k(m + n) \ll m.n$$

Matrix-vector multiplication: $Mx = y$

1. $w \leftarrow B^T x$
2. $y \leftarrow Aw$

- $O(k(m + n))$ instead of $O(mn)$

$\Rightarrow$ We will always use this representation for low-rank matrices
Is the SVD the only way to compute a low-rank approximation?

- The truncated SVD gives the best low-rank approximation
- Computing the SVD is $O(rN^3)$ (where $r$ is the rank of the approximation)

Do we need the SVD?

- We use the SVD: $M = U\Sigma V$ to compute the approximation

$$M = AB^T$$

with $A = U\Sigma$ and $B = V$

- We need a factorization but we do not need all the properties of the SVD

$\Rightarrow$ We use the Adaptive Cross Approximation
Cross approximation

Principle of the SVD

- SVD requires all entries of a block to construct a block low-rank
- Savings if we use only a small part of the entries

Principle of Cross approximation (or skeleton approximation)

- Choose an appropriate small set of pivot columns and pivot rows
- $A$ approximated by a combination of these few rows and columns
Principle of the Cross Approximation

Fully-pivoted Cross Approximation

- $\mathcal{R}(k)$—approximation: $k$ steps of rank-one approximations

![Diagram showing the process of cross approximation]

Successive approximations applied to the remainder

$$A = B_k + R_k, \quad R_k = A - \sum_{\ell=1}^{k} u_{\ell} v_{\ell}^T$$

At iteration $k$:

- Find the pivot $(i^*, j^*)$ such that $(i^*, j^*) = \arg\max_{i,j} |(R_k)_{i,j}|$
- Compute vectors: $u_{k+1} := \frac{(R_k)_{i^* j^*}}{(R_k)_{i^* j^*}}, \quad v_{k+1} := (R_k)_{i^* j}$
- Update the approximation: $B_{k+1} = B_k + u_{k+1} v_{k+1}^T$
Fully-pivoted Cross Approximation

Initialization: \( R_0 = A, B_0 = 0 \)

for all \( k = 0, 1, 2, \ldots \) do

Find pivot \( (i_{k+1}, j_{k+1}) := \text{argmax}_{i,j} |(R_k)_{ij}| \)

Normalization \( \gamma_{k+1} = \left( R_{i_{k+1}, j_{k+1}} \right)^{-1} \)

if \( \gamma_{k+1} \neq 0 \) then

Compute column \( u_{k+1} := \gamma_{k+1} R_k e_{j_{k+1}} \)

Compute row \( v_{k+1} := R_k^T e_{i_{k+1}} \)

New residue: \( R_{k+1} = R_k - u_{k+1} v_{k+1}^T \)

New approximation: \( B_{k+1} = B_k + u_{k+1} v_{k+1}^T \)

else

Terminate algorithm with exact rank \( k - 1 \)

end if

end for

It requires \( O(rmn) \) steps to generate an approximation of rank \( r \)
Fully-pivoted Cross Approximation

Initialization: $R_0 = A$, $B_0 = 0$

for all $k = 0, 1, 2, \ldots$ do

Find pivot $(i_{k+1}^*, j_{k+1}^*) := \arg\max_{i,j} |(R_k)_{ij}|$

Normalization $\gamma_{k+1} = \left(R_{i_{k+1}^*, j_{k+1}^*}\right)^{-1}$

if $\gamma_{k+1} \neq 0$ then

Compute column $u_{k+1} := \gamma_{k+1} R_k e_{j_{k+1}^*}$

Compute row $v_{k+1} := R_k^T e_{i_{k+1}^*}$

New residue: $R_{k+1} = R_k - u_{k+1} v_{k+1}^T$

New approximation: $B_{k+1} = B_k + u_{k+1} v_{k+1}^T$

else

Terminate algorithm with exact rank $k - 1$

end if

end for

- It requires $O(rmn)$ steps to generate an approximation of rank $r$
- It requires all the entries of $A$ to compute the pivot indices
Example of fully-pivoted Cross Approximation

\[ A = \begin{bmatrix}
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
8 & 6 & 7 & 5 & 4 & 7 & 4 & 5 \\
10 & 7 & 8 & 7 & 5 & 9 & 5 & 6 \\
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
8 & 6 & 7 & 5 & 4 & 7 & 4 & 5 \\
10 & 7 & 8 & 7 & 5 & 9 & 5 & 6 \\
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
10 & 7 & 8 & 7 & 5 & 9 & 5 & 6
\end{bmatrix} \]
Example of fully-pivoted Cross Approximation

\[
A = \begin{bmatrix}
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
8 & 6 & 7 & 5 & 4 & 7 & 4 & 5 \\
10 & 7 & 8 & 7 & 5 & 9 & 5 & 6 \\
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
8 & 6 & 7 & 5 & 4 & 7 & 4 & 5 \\
10 & 7 & 8 & 7 & 5 & 9 & 5 & 6 \\
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
10 & 7 & 8 & 7 & 5 & 9 & 5 & 6
\end{bmatrix}
\]

\[
u_1 := (0.6, 0.8, 1, 0.6, 0.8, 1, 0.6, 1)
\]

\[
v_1 := (10, 7, 8, 7, 5, 9, 5, 6)
\]
Example of fully-pivoted Cross Approximation

\[ A = \begin{bmatrix}
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
7 & 8 & 7 & 5 & 9 & 5 & 6 \\
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
8 & 6 & 7 & 5 & 4 & 7 & 4 & 5 \\
10 & 7 & 8 & 7 & 5 & 9 & 5 & 6 \\
6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
10 & 7 & 8 & 7 & 5 & 9 & 5 & 6
\end{bmatrix} \]

\[ u_1 := (0.6, 0.8, 1, 0.6, 0.8, 1, 0.6, 1) \]
\[ v_1 := (10, 7, 8, 7, 5, 9, 5, 6) \]

\[ R_1 = \begin{bmatrix}
0 & -0.2 & 0.2 & -0.2 & 0 & -0.4 & 0 & 0.4 \\
0 & 0.4 & 0.6 & -0.6 & 0 & -0.2 & 0 & 0.2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -0.2 & 0.2 & -0.2 & 0 & -0.4 & 0 & 0.4 \\
0 & 0.4 & 0.6 & -0.6 & 0 & -0.2 & 0 & 0.2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -0.2 & 0.2 & -0.2 & 0 & -0.4 & 0 & 0.4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \]
Example of fully-pivoted Cross Approximation

\[ A = \begin{bmatrix}
  6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
  8 & 6 & 7 & 5 & 4 & 7 & 4 & 5 \\
 10 & 7 & 8 & 7 & 5 & 9 & 5 & 6 \\
  6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
  8 & 6 & 7 & 5 & 4 & 7 & 4 & 5 \\
 10 & 7 & 8 & 7 & 5 & 9 & 5 & 6 \\
  6 & 4 & 5 & 4 & 3 & 5 & 3 & 4 \\
 10 & 7 & 8 & 7 & 5 & 9 & 5 & 6 \\
\end{bmatrix} \]

\[ u_1 := (0.6, 0.8, 1, 0.6, 0.8, 1, 0.6, 1) \]

\[ v_1 := (10, 7, 8, 7, 5, 9, 5, 6) \]

\[ R_1 = \begin{bmatrix}
  0 & -0.2 & 0.2 & -0.2 & 0 & -0.4 & 0 & 0.4 \\
  0 & 0.4 & 0.6 & -0.6 & 0 & -0.2 & 0 & 0.2 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & -0.2 & 0.2 & -0.2 & 0 & -0.4 & 0 & 0.4 \\
  0 & 0.4 & 0.6 & -0.6 & 0 & -0.2 & 0 & 0.2 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & -0.2 & 0.2 & -0.2 & 0 & -0.4 & 0 & 0.4 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \]

\[ u_2 := (1/3, 1, 0, 1/3, 1, 0, 1/3, 0), \quad v_2 := (0, 0.4, 0.6, -0.6, 0, -0.2, 0, 0.2) \]
Example of fully-pivoted Cross Approximation

\[ R_2 = \begin{bmatrix}
0 & -1/3 & 0 & 0 & 0 & -1/3 & 0 & 1/3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1/3 & 0 & 0 & 0 & -1/3 & 0 & 1/3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1/3 & 0 & 0 & 0 & -1/3 & 0 & 1/3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \]

Check that we have \( A = u_1 v_1^T + u_2 v_2^T + u_3 v_3^T \) (because \( \text{rank}(A) = 3 \))
Example of fully-pivoted Cross Approximation

\[
R_2 = \begin{bmatrix}
0 & -1/3 & 0 & 0 & 0 & -1/3 & 0 & 1/3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1/3 & 0 & 0 & 0 & -1/3 & 0 & 1/3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1/3 & 0 & 0 & 0 & -1/3 & 0 & 1/3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[u_3 := (1, 0, 0, 1, 0, 0, 1, 0), \quad v_3 := (0, -1/3, 0, 0, 0, -1/3, 0, 1/3)\]

Check that we have \( A = u_1 v_1^T + u_2 v_2^T + u_3 v_3^T \) (because \( \text{rank}(A) = 3 \))
Lemma (Exact reproduction of rank $k$ matrices)

Let $A$ be a matrix of rank exactly $k$. Then the matrix $B_k$ is equal to $A$.

$$ B_k := \sum_{\ell=1}^{k} u_\ell v_\ell^T $$

⇒ If rank($A$)=$r$, the algorithm terminates in $r$ steps

- Fully pivoted algorithm: requires computation of all entries of $A$
- How to reduce the complexity?
- Maximizes $|A_{ij}|$ for only one of the pairs of indices
  ⇒ requires computation of only few of the original matrix entries
Principle of the Partially-pivoted Cross Approximation

How do we find the vectors $\mathbf{u}$ and $\mathbf{v}$ to update the approximation?

- Fully-pivoted: pivot is the largest entry in the residual
- Partially-pivoted: maximize only for 1 of the 2 indices (the other one is fixed) $\rightarrow$ only one row or one column is assembled

At iteration $k$: the row index $i^*$ is given

- Find the column index $j^*$ such that $j^* = \arg\max_j |(R_k)_{i^*j^*}|$
- Compute vectors: $\mathbf{u}_{k+1} := \frac{(R_k)_{i^*j^*}}{(R_k)_{i^*j^*}}$, $\mathbf{v}_{k+1} := (R_k)_{i^*j^*}$
- Update the approximation: $B_{k+1} = B_k + \mathbf{u}_{k+1}\mathbf{v}_{k+1}^T$
- Find the row index $i^*$ such that $i^* := \arg\max_i |(R_k)_{ij^*}|$
Algorithm of Partially-pivoted CA

Initialization: \( B_0 = 0 \) and \( i_1^* = 1 \) (first row)

for all \( k = 0, 1, 2, \ldots \) do

Generation of the row: \( a := A^T e_{i_{k+1}^*} \)

Row of the residual \( R_k^T e_{i_{k+1}^*} = a - \sum_{\ell=1}^{k} (u_\ell)_{i_{k+1}^*} v_\ell \)

Find the pivot column \( j_{k+1}^* := \arg\max_j |(R_k)_{i_{k+1}^* j}| \)

Normalization \( \gamma_{k+1} = \left((R_k)_{i_{k+1}^* j_{k+1}^*}\right)^{-1} \)

if \( \gamma_{k+1} \neq 0 \) then

Generation of the column: \( a := A e_{j_{k+1}^*} \)

Column of the residual \( R_k e_{j_{k+1}^*} = a - \sum_{\ell=1}^{k} (v_\ell)_{j_{k+1}^*} u_\ell \)

Find the pivot row \( i_{k+2}^* := \arg\max_i |(R_k)_{i j_{k+1}^*}| \)

New vectors \( u_{k+1} := \gamma_{k+1} R_k e_{j_{k+1}^*}, v_{k+1} := R_k^T e_{i_{k+1}^*} \)

New approximation: \( B_{k+1} = B_k + u_{k+1} v_{k+1}^T \)

else

Terminate algorithm with exact rank \( k - 1 \)

end if

end for
Exact reproduction of rank $k$ matrices

Complexity of Partially-pivoted CA $O(k^2(m + n))$ instead of $O(rmn)$

Approximant and residual are not computed explicitly nor stored
Exact reproduction of rank $k$ matrices

Complexity of Partially-pivoted CA $O(k^2(m + n))$ instead of $O(rmn)$
Approximant and residual are not computed explicitly nor stored

Lemma (Exact reproduction of rank $k$ matrices)

Let $A$ be a matrix of rank exactly $k$. Then the matrix $B_k$ is equal to $A$.

\[ B_k := \sum_{\ell=1}^{k} u_\ell v_\ell^T \]

$\Rightarrow$ If rank($A$)=$r$, the algorithm terminates in $r$ steps
Adaptive Cross Approximation (ACA)

\[ A = B_k + R_k \]

- Simple heuristic to compute a cross approximation based on successive approximations by rank-1 matrices
- Information shifted iteratively from residual to approximant

Can we determine the rank \( k \) adaptively for a given approximation accuracy \( \varepsilon \)?

- Fully-pivoted ACA: \( \| A - B_k \|_F \leq \varepsilon \| A \|_F \)
- Partially-pivoted ACA: \( A \) is not assembled: stagnation-based error estimator
  \[ \| u_k \|_2 \| v_k \|_2 \leq \varepsilon \| B_k \|_F \]
How to reduce the costs of the BEM?

\[ \int_{\Gamma} G(x - y)p(y)\,dS_y = -u^{inc}(x), \quad x \in \Gamma \]

- Helmholtz fundamental solution: 
  \[ G(x - y) = \frac{\exp(ik|x - y|)}{4\pi|x - y|} \]

BEM discretization \( \Rightarrow \) fully-populated system of linear equations

\[ Ax = b \]

- Assembly of the matrix system is \( O(N^2) \)
- Matrix-vector product is \( O(N^2) \)

How can we reduce the costs?

- The system matrix is not assembled: \((A)_{ij}\) is not computed
- The matrix-vector product is approximated

- FM-BEM: need the analytic expression of the Green’s function
- Algebraic separation of the variables and low-rank approximation

\[ A \approx UV^T \]
The truncated SVD gives the best low-rank approximation

\[ A_r = \sum_{i=1}^{r} U_i \Sigma_{ii} V_i^*. \]

SVD: \( O(rN^3) \) (where \( r \) is the rank of the approximation)

We use the Adaptative Cross Approximation

\[ A = B_k + R_k, \quad R_k = A - \sum_{\ell=1}^{k} u_\ell v_\ell^T \]
Outline

1. Principle of Boundary Integral Equations
2. Review of classical Boundary Element Methods
3. Principle of the Fast Multipole Method
4. Fast Multipole accelerated BEM
5. Algebraic fast BEMs based on low rank approximations
   - Principle of Low rank approximations
   - Determining low-rank approximations
   - $\mathcal{H}$-matrices based Solvers
Low-rank approximation for the BEM matrix?

- $\mathbf{A} \in \mathbb{R}^{N \times N}$ of rank $k$ is called low-rank if $2kN \ll N^2$
- If the BEM matrix is low-rank, we know how to derive the low-rank approximation in a fast way

Structure of the BEM matrix: $\mathbf{A}\mathbf{p} = \mathbf{b}$

- $N$ collocation and interpolation points
- Interpolation of the unknown $\mathbf{p}(\mathbf{y})$: $\phi$
- Integration on the reference triangle with $N_g$ Gauss points

$$
\begin{bmatrix}
  b_1 \\
  \vdots \\
  b_N
\end{bmatrix} = 
\begin{bmatrix}
  G(x_c, y_g) \\
  \vdots \\
  w_{N_g}
\end{bmatrix} 
\begin{bmatrix}
  w_1 \\
  \vdots \\
  w_{N_g}
\end{bmatrix} 
\begin{bmatrix}
  \phi_1(y_1) & \ldots & \phi_N(y_1) \\
  \vdots & \ddots & \vdots \\
  \phi_1(y_{N_g}) & \ldots & \phi_N(y_{N_g})
\end{bmatrix} 
\begin{bmatrix}
  p_1 \\
  \vdots \\
  p_N
\end{bmatrix}
$$

$\mathbf{A} = \mathbf{G}\phi$

The existence of low-rank approximations (if any) is related to $\mathbf{G}$. 
Low-rank approximation for the BEM matrix?

- $A \in \mathbb{R}^{N \times N}$ of rank $k$ is called low-rank if $2kN \ll N^2$
- If the BEM matrix is low-rank, we know how to derive the low-rank approximation in a fast way

**Structure of the BEM matrix:** $Ap = b$

- $N$ collocation and interpolation points
- Interpolation of the unknown $p(y)$: $\phi$
- Integration on the reference triangle with $N_g$ Gauss points

$$
\begin{bmatrix}
  b_1 \\
  \vdots \\
  b_N
\end{bmatrix} = 
\begin{bmatrix}
  G(x_c, y_g)
\end{bmatrix}
\begin{bmatrix}
  \phi_1(y_1)w_1 & \cdots & \phi_N(y_1)w_1 \\
  \vdots & \ddots & \vdots \\
  \phi_1(y_{Ng})w_{Ng} & \cdots & \phi_N(y_{Ng})w_{Ng}
\end{bmatrix}
\begin{bmatrix}
p_1 \\
  \vdots \\
p_N
\end{bmatrix}
$$

$$A = G\phi$$

The existence of low-rank approximations (if any) is related to $G$. 


Behavior of the BEM system matrix

Model problem: Kernel function \((G)_{ij} = G(x_i, y_j)\)

\[
G_k = \sum_{i=1}^{k} U_i \Sigma_{ii} V_i^*, \quad \varepsilon_k = \frac{||G - G_k||_F}{||G||_F}
\]

Truncated SVD of the kernel function \((N = 1000, [-1 : 1], k = 10\pi)\)

- Decay of the sing. values
- Error decay

\[
||G - G_k||_F^2 = \sum_{i=k+1}^{N} \sigma_i^2
\]

⇒ We cannot find a low-rank approximation of the complete matrix.
What if we cannot perform a low-rank approximation?

What makes the matrix to be full-rank?

- Singular kernel as $||x - y||$ tends to 0
- Diagonal blocks contain all singularities
- Off-diagonal blocks: separated target and source points

⇒ We look for a data-sparse approximation: are there submatrices that are low-rank?

What about the FMM?

- Analytic low-rank approximation: when $x$ and $y$ are well-separated
- Use of an octree to separate near and far contributions
Finding submatrices that are low-rank

Subdivision of the interval into two subintervals

Behavior of singular values: off-diagonal block $[-1, 0] \times [0, 1]$

Decay of the sing. values

Error decay

Low-rank approximation of the blocks

Entire matrix

1 level

2 levels

3 levels
Main features of fast algebraic BEM solvers

- Recursive low-rank representation of off-diagonal blocks
- 1D: clustering of points is easy by grouping points by intervals
- 3D realistic scenario: we need a more complicated data structure

Common points to perform fast matrix-vector products

- Separation of near and far-field: Hierarchical separation of space
- Direct evaluation of near-field blocks
- Low-rank approximations of sub-blocks: ACA or SVD

⇒ Main features of hierarchical matrices ($\mathcal{H}$-matrices)
Basic concepts of Hierarchical matrices (vector case)

- Classical block partitioning: fixed partitioning of \( I = \{1, \ldots, N\} \) into disjoints subsets \( P = \{I_j : 1 \leq j \leq k\} \) with \( I = \bigcup_{j=1}^{k} I_j \)
- We need coarse partitionings and fine partitionings: \( \mathcal{H} \)-tree

Definition (\( \mathcal{H} \)-tree)

\( I \) is an index set, \( T \) a tree. \( T \) is a \( \mathcal{H} \)-tree (based on \( I \)) if

(i) All vertices \( t \in T \) are subsets of \( I \).
(ii) \( I \in T \) (\( I \) is the root of \( T \)).
(iii) All vertices \( t \in T \) has more than one son.
(iv) If \( t \in T \) not a leave, \( \mathcal{S}(t) \) contains disjoints subsets of \( I \) and

\[ t = \bigcup_{s \in \mathcal{S}(t)} s \]

---

Basic concepts of Hierarchical matrices (matrix case)

- Traditional block partitioning of a matrix: \( P_2 = P \times P \)
- We need finer blocks close to the diagonal and coarser far away

General block partitioning of \( I \times I \): allows subsets of \( I \times I \)
\( \mathcal{H} \)-partitionings: hierarchical structure similar to vector case

Example: binary trees
- Depth\(= 0 \): \( P_2(I, T) = \{ I \times I \} \)
- Depth\(= 1 \): \( I \) has 2 sons \( I_1 \) and \( I_2 \),
  \( P_2(I, T) = \{ I_1 \times I_1; I_1 \times I_2; I_2 \times I_1; I_2 \times I_2 \} \)
- Depth\(> 1 \): consider the subtrees \( T_k = S^*(I_k) \) for \( k = 1, 2 \)
  \( P_2(I, T) = P_2(I_1, T_1) \cup \{ I_1 \times I_2 \} \cup \{ I_2 \times I_1 \} \cup P(I_2, T_2) \)

Definition (\(\mathcal{H} \)-matrices)

Let \(P_2\) be a block partitioning of \(I \times I\) and \(k \in \mathbb{N}\). The set of \(\mathcal{H}\)-matrices induced by \(P_2\) is

\[
\mathcal{M}_{\mathcal{H},k}(I \times I, P_2) := \{ M \in \mathbb{K}^{I \times I} : \text{each block } M^b, b \in P_2, \text{ satisfies } \text{rank}(M^b) \leq k \}.
\]

Definition (\(\mathcal{R}_k\)-matrices)

A matrix \(A\) is called a \(\mathcal{R}_k\)-matrices if \(\text{rank}(A) \leq k\).

Properties of \(\mathcal{H}\)-matrices

- Not sparse matrices (in the sense only few non-zero entries)
- But \textbf{data sparse} matrices: described by only few data

\(\Rightarrow\) Need of an admissibility condition
So far partitionings are done explicitly on a regular 1D domain.
How do we perform the block partitioning in an efficient manner?

We need to define the following three tools for a general case:

1. Clustering of the points in the geometry: Cluster tree
2. Subdivision of the BEM matrix: Block Cluster tree
3. Separation between near and far blocks: Admissibility condition
Overview of the algorithm

\[ \int_{\Gamma} G(x - y)p(y) \, dS_y = -u^{inc}(x), \quad x \in \Gamma \]

1. Reumbering of the unknowns (according to the geometry): each index \( i \in I \) in the cluster tree carry a position \( x_i \in \mathbb{R}^3 \)
2. Subdivision of the BEM matrix: Block Cluster tree
3. Admissibility condition to determine which blocks are low-rank

- Full matrices: standard matrix-vector pd.
- Low-rank matrices:
  \[ A x \approx UV^* x = U(V^* x) \]

Cluster tree of the domain

- Before to partition the BEM matrix: we partition the boundary of the domain (\(H\)-tree or cluster tree)
- Classical block partitioning: fixed partitioning of \(I = \{1, \ldots, N\}\) into disjoints subsets \(P = \{I_j : 1 \leq j \leq k\}\) with \(I = \bigcup_{j=1}^{k} I_j\)

Clustering of a cloud of points: geometry information

- **Cluster**: set of indices corresponding to points that are "close" in some sense
- **Leaf**: node without son
- **Binary tree**: max of 2 sons per cell
- **The root cell encloses all the points**
- Recursive subdivision until stopping criteria (min \# of points) is achieved

\[\Rightarrow\] There are various algorithms to build a cluster tree
- Using cells and a separation plane
- Using a bisection criterion
Comparison of two cluster trees

Separation Plane: Axis of subdivision (largest edge of the box)
- Median bisection: \# of points in each cell is constant
- Geometric bisection: cell is subdivided according to the geometry

Median

Geometric

Balanced tree: principal direction based bisection

- Center of mass of the cluster \( X \)
- Covariance matrix of the cluster
  \[
  C = \sum_{k=1}^{N} (x_k - X)(x_k - X)^T
  \]
- Eigenvalues & Eigenvectors of \( C \)
- Separation plane: through \( X \) and orthogonal to largest eigenvalue

What do we need next?

\[ \int_{\Gamma} G(\mathbf{x} - \mathbf{y}) p(\mathbf{y}) dS_y = -u^{inc}(\mathbf{x}), \quad \mathbf{x} \in \Gamma \]

Important tools:

- ✓ Renumbering of the unknowns (according to the geometry): each index \( i \in I \) in the cluster tree carry a position \( \mathbf{x}_i \in \mathbb{R}^3 \)
- (2) Subdivision of the BEM matrix: Block Cluster tree
- (3) Admissibility condition to determine which blocks are low-rank

- Points on the mesh are partitioned the with a \( \mathcal{H} \)–tree over \( I \)
- We need to find submatrices of \( \mathbf{A} \) which will be low-rank \( \rightarrow \) Block Cluster tree
Principle of the Block Cluster tree

- Vector components are indexed by $i \in I$
- Matrix entries have indices from $I \times I$
- Block-cluster tree: cluster tree for $I \times I$

**Definition:**

- Cluster tree: $T_1 = T(I)$ and Block-cluster tree: $T_2 = T(I \times I)$
- Mapping $\tau$: constructs $T_2$ in a unique way from $T_1$
- $T_2$ belongs to $T_1 \times T_1$

**Construction of the mapping $\tau$:** Start with $I \times I \in T_2$ and define the sons of $b = (t_1, t_2) \in T_2$ recursively by
  
  - $(s_1, s_2)$ with $s_1 \in S(t_1)$, $s_2 \in S(t_2)$ if $S(t_1) \neq \emptyset$ and $S(t_2) \neq \emptyset$
  - $(t_1, s_2)$ with $s_2 \in S(t_2)$ if $S(t_1) = \emptyset$ and $S(t_2) \neq \emptyset$
  - $(s_1, t_2)$ with $s_1 \in S(t_1)$ if $S(t_1) \neq \emptyset$ and $S(t_2) = \emptyset$
  - $S_2(b) = \emptyset$ if $S(t_1) = \emptyset$ and $S(t_2) = \emptyset$
Vector components are indexed by $i \in I$: partition the geometry.

Matrix entries have indices from $I \times I$.

Block-cluster tree: cluster tree for $I \times I$.

**Definition of the submatrices**

- $\sigma$ and $\tau$: 2 nodes of the cluster tree.

- On the matrix, $\sigma$ and $\tau$ are subsets of the DOFs (rows and columns).

- $(\sigma, \tau) \in T_2$ defines the submatrices.

\[
T_1 = T(I)
\]

\[
A_{\sigma \times \tau}
\]
What do we need next?

\[ \int_{\Gamma} G(x - y)p(y)\,dS_y = -u^{inc}(x), \quad x \in \Gamma \]

√ Renumbering of the unknowns (according to the geometry):
each index \( i \in I \) in the cluster tree carry a position \( x_i \in \mathbb{R}^3 \)

√ Subdivision of the BEM matrix: Block Cluster tree

(3) Admissibility condition to determine which blocks are low-rank

We need an automatic tool to determine the low-rank submatrices:

- The blocks should be as large as possible
- Computing explicitly the rank of the submatrices is too expensive

⇒ Determine \textit{a priori} if the submatrix has a low-rank approximation?

- The condition depends on the nature of the matrix
- We know that we deal with BEM discretization
- The condition depends on the kernel used
Admissibility criteria: asymptotically smooth kernels

Definition (kernel asymptotically smooth)
A kernel is called asymptotically smooth if there exist constants $c_1$, $c_2 > 0$ and $g \leq 0$ such that for any multi-index $\alpha \in \mathbb{N}_0^d$

$$\forall x \neq y \quad |\partial^\alpha_y K(x, y)| \leq c_1p!(c_2)^p(|x - y|)^{g-p}, \quad p = |\alpha|$$

- Smoothness increases as the distance $|x - y|$ increases

Admissibility condition for asymptotically smooth kernels
$D_X, D_Y$: convex hull of set of points $x, y$. If we impose the condition

$$\text{diam } D_Y \leq \eta \text{ dist}(D_X, D_Y), \quad 0 \leq \eta \leq (c_2d)^{-1}$$

we can write ($\varepsilon_p \rightarrow 0$ for $p \rightarrow \infty$)

$$K(x, y) = \sum_{k=0}^{N_p} g_k(x)h_k(y) + R_p(x, y), \quad |R_p(x, y)| \leq \varepsilon_p$$

The corresponding matrix can be approximated by a matrix of rank $N_p$
Admissibility criteria: in practice

- Helmholtz kernel is not asymptotically smooth
- In practice: use of condition for asymptotically smooth kernels

\[ f(\sigma, \tau) = \begin{cases} 1 & \text{if } \min\left(\text{diam}(\sigma), \text{diam}(\tau)\right) < \eta d(\sigma, \tau) \\ 0 & \text{otherwise} \end{cases} \]

- Diameter of a group too expensive: diameter of enclosing cell

\[ \text{diam}(\sigma) \leq \sqrt{\sum_{i=1}^{3} (x_{\text{max}}^{i} - x_{\text{min}}^{i})^2} \]

- Distance: distance between closest faces of enclosing cells

\[ d(\sigma, \tau) \geq \sqrt{\sum_{i=1}^{3} \left(\max(0, x_{\text{min,}\tau}^{i} - x_{\text{max,}\sigma}^{i})\right)^2 + \left(\max(0, x_{\text{min,}\sigma}^{i} - x_{\text{max,}\tau}^{i})\right)^2} \]

Condition computed in \( O(1) \) operations
Admissible $T_2$-Partitionings

$$f(t_1, t_2) = \begin{cases} 
1 & \text{if } \min\left(\text{diam}(t_1), \text{diam}(t_2)\right) < \eta.d(t_1, t_2) \\
0 & \text{otherwise}
\end{cases}$$

**Definition (Admissible Block)**

A block $(t_1, t_2) \in T_2$ is admissible if either $b$ is a leaf or

$$f(t_1, t_2) = 1.$$ 

**Definition (Admissible $T_2$-Partitionings)**

A $T_2$-Partitioning $P$ of $I \times I$ is admissible, if all blocks $t \in P$ are admissible.
Hierarchical algorithm: summary

✓ Renumbering of the unknowns (according to the geometry):
  each index \( i \in I \) in the cluster tree carry a position \( x_i \in \mathbb{R}^3 \)

✓ Subdivision of the BEM matrix: Block Cluster tree

✓ Admissibility condition to determine which blocks are low-rank

- Better compression between large, well separated blocks
- If blocks are too large: more non admissible blocks

Use of a hierarchical algorithm (similarly to the FMM)
- Leaves of the block cluster tree
  - if blocks are not admissible: store in a component-wise fashion
  - otherwise admissibility condition guarantees low-rank sub-block
- Non leaf blocks
  - if blocks are not admissible: the block is subdivided
  - otherwise admissibility condition guarantees low-rank sub-block
- Iterations until admissible block or the stopping criteria
- 3 kinds of blocks: \( \mathcal{H} \)-matrices; full- and low-rank blocks

Advantages of hierarchical matrices

Blockwise low-rank approximations of the matrix
- Black-box construction of cluster trees and block cluster trees
- Black-box construction of low-rank approximations
- Approximate matrix arithmetic operations

Solvers for hierarchical matrices: algebraic approach
1. Matrix-vector multiplication needed for iterative solvers
2. LU factorization needed for a direct solver
Only operation needed for the iterative solver: matrix-vector product

\[ y = Ax := \text{MVM}(A, \tau \times \sigma, x, y) \]

\[
\text{if } \tau \times \sigma \text{ not a leaf cell then}
\]
\[
\text{for all } \tau' \times \sigma' \in S(\tau \times \sigma) \text{ do}
\]
\[
\text{Matrix-vector product: } \text{MVM}(A, \tau' \times \sigma', x, y)
\]
\[
\text{end for}
\]
\[
\text{else}
\]
\[
\text{end if}
\]

\[
y_{|\tau} := y_{|\tau} + A_{\tau \times \sigma} x_{|\sigma}
\]
Matrix-vector product involving a $\mathcal{H}$– matrix

Only operation needed for the iterative solver: matrix-vector product

$$y = Ax := \text{MVM}(A, \tau \times \sigma, x, y)$$

if $\tau \times \sigma$ not a leaf cell then
  for all $\tau' \times \sigma' \in S(\tau \times \sigma)$ do
    Matrix-vector product: \text{MVM}(A, \tau' \times \sigma', x, y)
  end for
else
  $y|_{\tau} := y|_{\tau} + A_{\tau \times \sigma}x|_{\sigma}$
end if

Matrix-vector product for the leaf cells

- If it is a full-block: standard matrix-vector product
- If it is an admissible block: $A_{\tau \times \sigma} = UV^T$
  1. $w \leftarrow V^T x|_{\sigma}$
  2. $y|_{\tau} \leftarrow Uw$

$\Rightarrow$ Easy Implementation of an iterative solver for $\mathcal{H}$–matrices
LU-solver for 2x2 Block System

\[
\begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}
\]

We introduce \( Y = UX \), i.e. \( LY = B \). LU-solver decomposed into

1. **Forward step:** solve \( LY = B \)

\[
\begin{align*}
L_{11}y_1 &= b_1 \\
L_{22}y_2 &= b_2 - L_{21}y_1
\end{align*}
\]

2. **Backward step:** solve \( Y = UX \)

\[
\begin{align*}
x_2 &= U_{22}^{-1}y_2 \\
x_1 &= U_{11}^{-1}y_1 - U_{11}^{-1}U_{12}x_2
\end{align*}
\]
Principle of $\mathcal{H}$-LU factorization

- $L$ (resp. $U$): lower (resp. upper) triangular $\mathcal{H}$-matrices

\[ A \approx LU \]

Hierarchical block structure: recursion on $2 \times 2$ block matrices

- $(\tau, \tau) \in T(I \times I) \backslash L(T(I \times I)) \ (\tau_1$ and $\tau_2$: sons of $\tau)$

\[
\begin{pmatrix}
A_{\tau_1 \tau_1} & A_{\tau_1 \tau_2} \\
A_{\tau_2 \tau_1} & A_{\tau_2 \tau_2}
\end{pmatrix}
= \begin{pmatrix}
L_{\tau_1 \tau_1} & 0 \\
\mathbf{0} & L_{\tau_2 \tau_2}
\end{pmatrix}
\begin{pmatrix}
U_{\tau_1 \tau_1} & U_{\tau_1 \tau_2} \\
\mathbf{0} & U_{\tau_2 \tau_2}
\end{pmatrix}
\]

Decomposition into four subproblems

\[
\begin{pmatrix}
A_{\tau_1\tau_1} & A_{\tau_1\tau_2} \\
A_{\tau_2\tau_1} & A_{\tau_2\tau_2}
\end{pmatrix} =
\begin{pmatrix}
L_{\tau_1\tau_1} & 0 \\
L_{\tau_2\tau_1} & L_{\tau_2\tau_2}
\end{pmatrix}
\begin{pmatrix}
U_{\tau_1\tau_1} & U_{\tau_1\tau_2} \\
0 & U_{\tau_2\tau_2}
\end{pmatrix}
\]

1. LU decomposition to compute: \(L_{\tau_1\tau_1}\) and \(U_{\tau_1\tau_1}\)
2. Compute \(U_{\tau_1\tau_2}\) from \(A_{\tau_1\tau_2} = L_{\tau_1\tau_1} U_{\tau_1\tau_2}\)
3. Compute \(L_{\tau_2\tau_1}\) from \(A_{\tau_2\tau_1} = L_{\tau_2\tau_1} U_{\tau_1\tau_1}\)
4. LU decomposition to compute: \(A_{\tau_2\tau_2} - L_{\tau_2\tau_1} U_{\tau_1\tau_2} = L_{\tau_2\tau_2} U_{\tau_2\tau_2}\)

Needed tools:

- LU solver: recursive function
- Efficient solver for lower and upper triangular matrices

Use of the hierarchical structure for $\mathcal{H}$-LU factorization

\[ LU(A) \]

if the block $A$ is a leaf then
   $A$ is a full matrix, use standard LU decomposition
else

\[
\begin{pmatrix}
A_{τ_1τ_1} & A_{τ_1τ_2} \\
A_{τ_2τ_1} & A_{τ_2τ_2}
\end{pmatrix} =
\begin{pmatrix}
L_{τ_1τ_1} & 0 \\
L_{τ_2τ_1} & L_{τ_2τ_2}
\end{pmatrix}
\begin{pmatrix}
U_{τ_1τ_1} & U_{τ_1τ_2} \\
0 & U_{τ_2τ_2}
\end{pmatrix}
\]

$LU(A_{τ_1τ_1})$
Solve $A_{τ_1τ_2} = L_{τ_1τ_1} U_{τ_1τ_2}$
Solve $A_{τ_2τ_1} = L_{τ_2τ_1} U_{τ_1τ_1}$
Schur Complement $A_{τ_2τ_2} - L_{τ_2τ_1} U_{τ_1τ_2}$

$LU(A_{τ_2τ_2} - L_{τ_2τ_1} U_{τ_1τ_2})$
end if

⇒ Efficient solver for lower and upper triangular matrices?
Recursive block forward substitution

\[ \text{BFS}(A, L) \]

**if** the block \( A \) is a leaf **then**

Use standard forward substitution

**else**

\[
\begin{pmatrix}
A_{\tau_1 \tau_1} & A_{\tau_1 \tau_2} \\
A_{\tau_2 \tau_1} & A_{\tau_2 \tau_2}
\end{pmatrix}
= \begin{pmatrix}
L_{\tau_1 \tau_1} & 0 \\
L_{\tau_2 \tau_1} & L_{\tau_2 \tau_2}
\end{pmatrix}
\begin{pmatrix}
B_{\tau_1 \tau_1} & B_{\tau_1 \tau_2} \\
B_{\tau_2 \tau_1} & B_{\tau_2 \tau_2}
\end{pmatrix}
\]

\[
B_{\tau_1 \tau_1} = \text{BFS}(A_{\tau_1 \tau_1}, L_{\tau_1 \tau_1})
\]

\[
B_{\tau_1 \tau_2} = \text{BFS}(A_{\tau_1 \tau_2}, L_{\tau_1 \tau_1})
\]

\[
B_{\tau_2 \tau_1} = \text{BFS}(A_{\tau_2 \tau_1} - L_{\tau_2 \tau_1} B_{\tau_1 \tau_1}, L_{\tau_2 \tau_2})
\]

\[
B_{\tau_2 \tau_2} = \text{BFS}(A_{\tau_2 \tau_2} - L_{\tau_2 \tau_1} B_{\tau_1 \tau_2}, L_{\tau_2 \tau_2})
\]

**end if**

\[ \Rightarrow \text{Similar approach for upper triangular systems} \]